SIMUCAD

Expert

Layout Editor

ExpertViews

Layout Viewer

User's Manual

Simucad
4701 Patrick Henry Drive, Bldg. 2
Santa Clara, CA 95054
Telephone (408) 567-1000
Internet http://www.simucad.com

May 8, 2008

Expert Layout Editor
and
ExpertViews Layout Viewer
User's Manual
Copyright 2008

Simucad
4701 Patrick Henry Drive, Building #2
Santa Clara, CA 95054

Phone: (408) 567-1000

Notice

The information contained in this document is subject to change without notice.

SIMUCAD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE.

Simucad shall not be liable for errors contained herein, or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

This document contains proprietary information, which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated into another language without the prior written consent of Simucad.

GUARDIAN, GUARDIAN DRC, REALTIME DRC, GUARDIAN LVS, GUARDIAN NET, EXPERT, EXPERTVIEWS, GATEWAY, HARMONY, SMARTSPICE, SMARTSPICERF, SMARTSPICESEE, SMARTVIEW, SMARTLIB, SDDL, TWISTER, SPRINT, HARMONY, SILOS, HYPERFAULT, TURBOLINT, LISA, QUEST, EXACT, CLEVER, STELLAR, HIPEX-C, HIPEX-CRC, HIPEX-NET, HIPEX-RC, NOMAD, SCOUT, EDA OMNI, UTMOST, UTMOST III, UTMOST IV, SPAYN, ACCUCELL, ACCUCORE, ACCUMODEL, ACCUTEST, SFLM, VYPER, INTERPRETER, DECKBUILD, SMARTLIB, CIRCUIT OPTIMIZER, PROMOST, RESILIANCE, DISCOVERY, ANALOG EXPRESS, ANALOG IC DESIGN FLOW, CELEBRITY, CELEBRITY_C++ are trademarks of Simucad.

© 1990-2008 by Simucad Design Automation, Inc.

Style Conventions		
Font Style/Convention	**Description**	**Example**
•	This represents a list of items or terms.	• Bullet A • Bullet B • Bullet C
1. 2. 3.	This represents a set of directions to perform an action.	To open a door: 1. Unlock the door by inserting the key into keyhole. 2. Turn key counter-clockwise. 3. Pull out the key from the keyhole. 4. Grab the doorknob and turn clockwise and pull.
→	This represents a sequence of menu options and GUI buttons to perform an action.	**File→Open**
`Courier`	This represents the commands, parameters, and variables syntax.	HAPPY BIRTHDAY
New Century Schoolbook Bold	This represents the menu options and buttons in the GUI.	File
New Century Schoolbook Italics	This represents the equations.	*abc=xyz*
Note:	This represents the additional important information.	**Note:** Make sure you save often while running an experiment.
New Century SchoolBook in Small Caps	This represents the names of the products.	ATHENA, ATLAS, EXPERT, GATEWAY, HIPEX, SMARTSPICE, STELLAR, and UTMOST.

Table of Contents

Chapter 1:
Introduction 1-1

1.1: About This Manual 1-1
1.2: Overview of Expert Capabilities 1-1
1.2.1: Advanced Capabilities 1-2
1.2.2: Summary of Features 1-3
1.2.3: Layout Database Handling 1-5

Chapter 2:
Basic Notions 2-1

2.1: Projects, Cells, Libraries 2-1
2.1.1: Cell Name Restriction 2-2
2.2: Hierarchical IC Design 2-2
2.2.1: Dealing with Recursive Instancing 2-2
2.2.2: Parameterized Cells 2-3
2.3: IC Layout Design Concepts 2-3
2.3.1: Design Plane 2-3
2.3.2: Layout Objects 2-4
2.3.3: User Defined Properties 2-6
2.4: Editor Accessories 2-7
2.5: Files 2-7
2.5.1: Expert Layout Database (*.eld file) 2-7
2.5.2: Shared Expert Layout Database (*.sld file) 2-8
2.5.3: GDSII file (*.gds file) 2-8
2.5.4: CIF file (*.cif file) 2-8
2.5.5: Applicon file (*.apl file) 2-8
2.5.6: OASIS file(*.oas) 2-8
2.5.7: LEF/DEF file(*.lef, *.def) 2-8
2.5.8: Technology files (*.tcn file) 2-8
2.5.9: Other Expert Files 2-12

Chapter 3:
User Interface 3-1

3.1: Expert Interface 3-1
3.1.1: Menus 3-2
3.1.2: Toolbars and Dock Windows 3-9
3.1.3: Mouse/Cursor Functions 3-10
3.1.4: Dialogs 3-12
3.2: Command Execution 3-12
3.2.1: Breaking Execution 3-13
3.2.2: Running Previous Command 3-13
3.3: Entering and Exiting Expert 3-13
3.4: Customization 3-14
3.4.1: Shortcuts 3-14
3.4.2: Color Setup 3-14
3.4.3: Custom Pop-up Menu 3-14

3.4.4: Simple Custom Submenu 3-15

Chapter 4:
Project/Cell Commands 4-1

4.1: Overview. 4-1
4.2: †Starting New Project 4-3
4.2.1: Technology File Selection 4-4
4.3: Loading Project 4-5
4.3.1: Open... Command 4-5
4.3.2: Most Recent Used Project List 4-6
4.3.3: Loading from the ELD format 4-6
4.3.4: Loading from the SLD format 4-7
4.3.5: Loading from a Foreign Format 4-7
4.3.6: Loading from LEF or DEF Format 4-7
4.3.7: Cell Selection Dialog 4-7
4.3.8: Possible Problems 4-9
4.4: Saving/Compressing the Project 4-10
4.4.1: Save Cells 4-10
4.4.2: Save 4-10
4.4.3: Save As 4-11
4.4.4: Compress 4-11
4.5: Closing Project 4-11
4.6: †Cell->New 4-12
4.6.1: Possible Problems 4-12
4.7: Opening Cells 4-13
4.7.1: Open.... 4-13
4.7.2: Descend 4-13
4.7.3: Ascend. 4-13
4.7.4: Possible Problems 4-13
4.8: †Creating a New Cell from Selected Objects 4-14
4.9: †Copying of Cells 4-15
4.9.1: Cell Name Collision Resolving Dialog 4-16
4.10: †Deleting Cells 4-17
4.11: †Renaming of Cells 4-18
4.12: Importing Cells to Project 4-18
4.13: †Exporting Cells 4-19
4.13.1: Cell Export... to GDSII Files 4-20
4.13.2: Cell Export... to Xi-Script Files 4-20
4.14: Saving /Discarding/Forbidding Cell Changes 4-21
4.14.1: Save 4-21
4.14.2: Save As... 4-21
4.14.3: Discard Changes 4-21
4.14.4: Read Only 4-21
4.15: Cell Information 4-21

Chapter 5:
Creating Layout Objects 5-1

5.1: Creating Layouts 5-1
5.2: Drawing Settings 5-2
5.2.1: Admissible Angles 5-2
5.2.2: Snap Cursor to Grid 5-2
5.2.3: Direction Snap Modes 5-2
5.2.4: Region Modes 5-3
5.3: Cell Instance/Array Creation 5-6
5.3.1: Instance/Array Attributes 5-6
5.3.2: Instance of Parameterized Cell 5-8
5.4: Primitive Object Creation 5-9
5.4.1: Polygonal Region Creation 5-9
5.4.2: Box Region Creation 5-10
5.4.3: Circle/Ellipse Creation 5-11
5.4.4: Donut (Circular Ring) Shape Creation 5-12
5.4.5: Rolled Region Creation 5-12
5.4.6: Wire Creation 5-13
5.4.7: Rolled Wire Creation 5-15
5.5: Text/Label Creation 5-15
5.6: Parametric Wire 5-16
5.7: Contact 5-21
5.8: Bus 5-23

Chapter 6:
Modifying Layout Objects 6-1

6.1: Objects 6-1
6.2: General 6-1

Chapter 7:
Selecting Layout Objects 7-1

7.1: Layout Objects 7-1
7.2: Select By Click/Drag 7-1
7.2.1: Selection Modes 7-2
7.2.2: Selection Among Overlapping Objects 7-2
7.2.3: Preselection Mode 7-3
7.3: Box Selection 7-4
7.4: List Selection 7-5
7.5: Deselection 7-5
7.6: Save Selected As Cell 7-6
7.7: Save Selected Copy To Active Layer 7-6
7.8: Save Selected Move To Layer 7-7
7.9: Other Selection Commands/Options 7-7
7.9.1: All 7-7
7.9.2: Active Layer 7-7
7.9.3: Selection by Object Type 7-7

7.9.4: Select Newest 7-7
7.9.5: Invert Selection 7-8
7.9.6: Cell Instance Selectivity. 7-8
7.9.7: Primitive Object Selectivity 7-8
7.9.8: Selection Count Indicator 7-8
7.9.9: Selection Highlight Modes 7-8
7.9.10: Select Newest Objects 7-8
7.9.11: Selection Tool as Default. 7-8
7.9.12: Fat Preselection. 7-8
7.9.13: Layer Selectability/Visibility 7-8
7.9.14: Setup Hotkeys/Menus/Colors... Color 7-8

Chapter 8:
Other Edit Submenu Operations 8-1

8.1: Editing Commands 8-1
8.2: Expert Clipboard 8-3
8.2.1: Cut 8-3
8.2.2: Copy 8-3
8.2.3: Paste 8-3
8.2.4: Paste into Layer. 8-4
8.3: Delete 8-4
8.4: Position Cursor 8-4
8.5: Geometric Transformations 8-5
8.5.1: Edit Duplicate, Edit Move 8-5
8.5.2: Rotate-90. 8-7
8.5.3: Rotate 8-7
8.5.4: Mirroring Operations 8-8
8.5.5: Set Origin. 8-8
8.6: Undo/Redo 8-9
8.6.1: Esc Key Interrupts Editing Operations. 8-9
8.6.2: Undo 8-9
8.6.3: Redo 8-9
8.7: Stretch Operation 8-10
8.7.1: Overview 8-10
8.7.2: Sample Run. 8-11
8.7.3: Specification 8-11
8.8: Jog Points 8-12
8.9: Inspecting/Changing Object Properties 8-13
8.10: Searching Layout Objects 8-15
8.11: Other Editing Commands 8-18
8.12: Edit Submenu Operations 8-19
8.12.1: Stitch Wires 8-19
8.12.2: Align. 8-20

Chapter 9:
Viewing the Layout. 9-1

9.1: Layout Windows 9-1

9.1.1: Opening/Closing Windows 9-1
9.1.2: Window Arrangements 9-1
9.2: Filling Modes 9-3
9.2.1: Individual Filling Modes for Layers 9-5
9.2.2: Individual Filling Modes for the Active Layer 9-5
9.3: Panning Commands. 9-6
9.3.1: Pan To Target Point 9-7
9.3.2: Pan To Cursor Position 9-7
9.3.3: Pan Scroll Left/Right/Up/Down 9-7
9.3.4: Pan To Reference Point 9-7
9.3.5: Pan Autopan. 9-7
9.4: Zoom 9-8
9.4.1: Zoom to Fit 9-8
9.4.2: Zoom Selected 9-8
9.4.3: Zoom Into Window 9-8
9.4.4: Zoom Out of Window 9-9
9.4.5: Fixed-Ratio Zoom In and Out 9-10
9.5: Mouse Wheel for Scrolling and Zooming. 9-10
9.6: Cell Map. 9-10
9.7: View Cell View 9-11
9.7.1: Flat 9-11
9.7.2: Lazy 9-12
9.7.3: Floorplan Mode 9-13
9.7.4: Primitives Mode 9-13
9.7.5: Edited Cell Frame. 9-13
9.7.6: Hierarchy Depth of Display. 9-13
9.7.7: Activate Layout View 9-13
9.8: View Array View 9-14
9.9: Layer View. 9-14
9.9.1: Ordering 9-14
9.9.2: Layer Plan 9-15
9.10: Temporary Reference Point 9-15
9.10.1: Toggle Reference Point 9-15
9.10.2: Activate Reference Point 9-15
9.10.3: Traveling Reference Point 9-15
9.10.4: To Reference Point 9-16
9.10.5: Distance from Reference Point 9-16
9.11: Various 9-16
9.11.1: Redraw 9-16
9.11.2: Redraw Region. 9-16
9.11.3: Cursor Crosshairs 9-16
9.11.4: Show Grid. 9-16
9.11.5: Dock Windows 9-16
9.11.6: Toolbars 9-16

Chapter 10: Hierarchy Operations 10-1

10.1: Hierarchy Menu 10-1
10.2: Hierarchy Navigator (Project Tree) 10-1
10.2.1: Project Tree 10-1
10.3: Edit-in-Place 10-3
10.3.1: Enter into Subcell 10-3
10.3.2: Direct Pick of Object in Hierarchy 10-5
10.3.3: Up 10-5
10.3.4: Exit 10-6
10.3.5: End Editing Dialog 10-6
10.3.6: EIP Highlighting 10-7
10.4: Create Cell in-Place 10-7
10.5: Explode Selected 10-8
10.6: Explode All 10-8
10.7: Flatten 10-9
10.8: Replacing Cell Instances 10-9

Chapter 11: Setup 11-1

11.1: Setting Up Expert&ExpertViews 11-1
11.2: Managing Setups 11-2
11.2.1: Technology Settings 11-2
11.2.2: Security of Technology Settings 11-3
11.2.3: Editor Preferences 11-4
11.3: Editor/Viewer Form 11-5
11.3.1: General Page 11-5
11.3.2: Viewing Page 11-6
11.3.3: Editing Page 11-8
11.3.4: Data Safety Page 11-10
11.3.5: Directories and Files Page 11-11
11.3.6: Ruler Page 11-12
11.3.7: Infix Page 11-12
11.3.8: Gravity Page 11-13
11.4: Technology Setup 11-14
11.4.1: General Page 11-14
11.4.2: Shapes Page 11-15
11.4.3: Layer/Wire Setup 11-15
11.4.4: Connectivity Setup 11-29
11.4.5: Device Setup 11-29
11.4.6: Grid Setup 11-29
11.4.7: Pwire Templates 11-32
11.4.8: Contact Setup 11-33
11.4.9: Instance Grid 11-34
11.5: GDSII I/O Setup 11-35
11.5.1: Description of Particular Options 11-36

11.6: CIF I/O Setup 11-41
11.7: Applicon I/O Setup 11-42
11.8: OASIS I/O Setup 11-43
11.9: Shortcut, Custom Menu, Custom Color Setups 11-44
11.9.1: Customizable Hotkeys 11-44
11.9.2: Toolbars Panel 11-45
11.9.3: Application Colors Panel 11-46
11.9.4: Custom Menu 11-46
11.10: Built-in DRC Command Setup 11-49
11.11: DRC Script Run Setup 11-50

Chapter 12:
Tools 12-1

12.1: Layout Tools 12-1
12.2: Measurement 12-2
12.2.1: Metric Bar 12-2
12.2.2: Polyline Ruler 12-4
12.2.3: Coordinate Axes 12-5
12.2.4: Coordinate Origin 12-5
12.2.5: Cursor Crosshairs 12-5
12.2.6: Distance from Point to Line 12-5
12.2.7: Distance from Reference Point 12-6
12.3: †Xi Script Submenu 12-6
12.4: †NDL/Netlist Driven Layout 12-7
12.4.1: Loading the Netlist 12-7
12.4.2: Cell Creation 12-9
12.4.3: Layout Editing 12-10
12.4.4: NDL Submenu 12-10
12.4.5: Cells/P-cells Requirements 12-12
12.5: Cross Sectional Viewer 12-14
12.5.1: Introduction 12-14
12.5.2: Cross Sectional Viewer Setup 12-14
12.5.3: Using the Cross Sectional Viewer 12-20
12.6: Device Linking 12-23
12.6.1: Linking Setup 12-23
12.6.2: Automatic Linking 12-24
12.6.3: Manual Linking 12-26
12.7: †Cut by (Poly)Line 12-27
12.8: Cut by Vertices Count 12-28
12.9: Split Compound Objects 12-29
12.10: †Clip Out/Clip In 12-30
12.11: †Merge Selected 12-32
12.12: †Resize Selected 12-32
12.13: Technology Migration 12-34
12.13.1: †Scale Selected and Scale Whole Layout 12-34
12.13.2: †Snap Selected (or Whole Layout) to Grid 12-35

12.14: Handling Derived Layers 12-36
12.15: Merge Two Wires 12-37
12.16: Various 12-38
12.16.1: Scratch Layers 12-38
12.16.2: Pick Layer 12-38
12.16.3: Polygonize 12-38
12.16.4: Mouse Click 12-38
12.16.5: Attach Texts 12-38
12.16.6: Detach Text 12-38
12.16.7: Previous Tool 12-38
12.16.8: Void Tool 12-38

Chapter 13:
Parameterized Cells 13-1

13.1: Overview 13-1
13.2: P-cell Menu Command 13-2
13.3: Creating graphical P-cells 13-3
13.3.1: Graphical P-cell Creation Flow 13-3
13.3.2: Defining Graphical P-cell Rules 13-5
13.3.3: Defining Graphical P-cell Parameters 13-9
13.3.4: Graphical P-cell Preface 13-10
13.3.5: Graphical P-cell Compiling 13-10
13.4: Creating Textual P-cells 13-11
13.4.1: Textual P-cell Creation Flow 13-11
13.5: Conversion From Graphical to Textual P-cell 13-12
13.6: P-cell Password Protection 13-13

Chapter 14:
Integrated Layout Verification Tools 14-1

14.1: Layout Verification Tools 14-1
14.2: Geometrical DRC System 14-2
14.2.1: Basics of Usage in Expert 14-2
14.2.2: Real-time DRC 14-5
14.3: Guardian LPE Netlist Extractor 14-7
14.4: Guardian LVS Netlist Comparator 14-9
14.5: Node Probing 14-10
14.5.1: Overview 14-10
14.5.2: Short Locator 14-12
14.5.3: Node Probing Submenu 14-13
14.6: Calibre 14-14

Chapter 15:
Plotting 15-1

15.1: Plotting Layouts 15-1
15.2: Printer Setup Dialog 15-2
15.3: Print Preview 15-3
15.4: Page Setup Dialog 15-4

15.4.1: Page Layout Sheet 15-4
15.4.2: Plotting Style Sheet 15-6
15.4.3: Miscellaneous Sheet 15-9
15.5: Print 15-10
15.6: Print Status 15-11

Chapter 16:
Libraries and Version Control for a Multi-User Design Environment 16-1

† 16.1: Overview 16-1
† 16.2: Libraries 16-1
† 16.3: Typical Scenarios for Library Usage 16-2
16.3.1: Non-Shared Libraries 16-2
16.3.2: Shared Libraries/Cell Version Control 16-2
16.3.3: Shared Access to Projects in Read-Only Mode 16-2
† 16.4: Project/Library Management 16-3
16.4.1: Library Setup for Your Project 16-3
16.4.2: Library and Cell Access Rights 16-5
† 16.5: Cell Management 16-7
16.5.1: Usage of Library Cells in Projects 16-7
16.5.2: Cell Version Management 16-7
16.5.3: Edited Cell Version Management 16-8

Chapter 17:
Expert Interface Scripts (XI-Shell) 17-1

17.1: Scripting Language (LISA) for Expert Commands 17-1
17.2: Remarks on Command Syntax and Descriptions 17-2
17.3: Project Commands and Procedures 17-3
17.3.1: Project Commands 17-3
17.3.2: Project Procedures 17-5
17.4: Cell Commands and Procedures 17-8
17.4.1: Cell Commands 17-8
17.4.2: Cell Procedures 17-16
17.5: Cell View Commands and Procedures 17-19
17.5.1: Cell View Commands 17-19
17.5.2: Cell View Procedures 17-21
17.6: Setup Commands 17-21
17.6.1: Set Directories Commands 17-21
17.6.2: Expert Settings Commands 17-22
17.6.3: Expert Setup Commands 17-23
17.6.4: XI Setup Commands 17-27
17.6.5: Edit Settings Commands 17-28
17.6.6: Getting/Releasing Simucad Licenses Commands 17-30
17.7: Layer Commands and Procedures 17-31
17.7.1: Layer Commands 17-31
17.7.2: Layer Procedures 17-36
17.8: Commands for Creating Objects 17-38

17.8.1: Common Named Arguments and Switches. . . . 17-38
17.8.2: Primitive Object Creation. . . . 17-40
17.8.3: Instance Object Creation. . . . 17-49
17.8.4: Device Instance Object Creation. . . . 17-53
17.8.5: Floorplan Object Creation 17-54
17.8.6: Complex Object Creation 17-57
17.9: Select Commands 17-61
17.10: Edit Commands 17-65
17.11: Search Commands and Procedures 17-70
17.12: Modify Object Commands 17-72
17.12.1: Common Arguments and Named Arguments 17-72
17.12.2: Modify Commands. . . . 17-72
17.13: Tools Commands 17-76
17.14: Hierarchy Commands 17-83
17.15: Edit-in-place Commands 17-84
17.16: Connectivity Commands 17-85
17.17: Verification Commands 17-87
17.18: Library/User Management Commands and Procedures 17-90
17.18.1: Library Setup Commands 17-90
17.18.2: Library Management Procedures 17-92
17.18.3: Cell Management Commands. . . . 17-93
17.19: Miscellaneous 17-94
17.20: EXPERT/LISA Data Types. . . . 17-104
17.20.1: POINT Type. . . . 17-104
17.20.2: RECT Type 17-105
17.20.3: LAYOUTOBJECT Type. . . . 17-106
17.20.4: SEARCHCRITERION Type. . . . 17-109
17.20.5: OBJECTCONTAINER Type 17-110
17.20.6: EXPERT/LISA Data Type Procedures 17-110
17.20.7: EXPERT_SETUP Type. . . . 17-115
17.21: Creating P-cells command 17-117
17.22: Access to Xi-commands. . . . 17-120
17.22.1: Command Line Input. . . . 17-120
17.22.2: Xi Script Panel. . . . 17-120
17.22.3: Activity Log 17-121
17.22.4: CellÆExportÆXi-script 17-121
17.23: Command Quick Reference 17-121

Chapter 18:
Expert API 18-1

18.1: Overview. . . . 18-1
18.1.1: Plug-in Libraries. . . . 18-1
18.1.2: Structure of Objects in Expert API Library 18-1
18.1.3: Access to Functions from Plug-in Library 18-1
18.2: Access to Expert Settings 18-2
18.2.1: Save Settings into File. . . . 18-2

18.2.2: Load Settings from File 18-2
18.2.3: Set Measurement Units 18-3
18.2.4: Get Measurement Units 18-3
18.2.5: Get Measurement Unit Value in Microns 18-3
18.2.6: Set Drawing Angle Mode 18-4
18.2.7: Get Angle Mode 18-4
18.2.8: Set Region Mode 18-4
18.2.9: Get Region Mode 18-5
18.2.10: Set General Setup Data 18-5
18.2.11: Get General Setup Data 18-5
18.2.12: Set Technology Setup Data 18-6
18.2.13: Get Technology Setup Data 18-6
18.2.14: Set GDSII Setup Data 18-6
18.2.15: Get GDSII Setup Data 18-7
18.2.16: Set CIF Setup Data 18-7
18.2.17: Get CIF Setup Data 18-7
18.2.18: Set Applicon Setup Data 18-8
18.2.19: Get Applicon Setup Data 18-8
18.3: Project 18-8
18.3.1: Is Project Open 18-8
18.3.2: New Project 18-9
18.3.3: Load Project 18-9
18.3.4: Close Project 18-10
18.3.5: Save Project 18-10
18.3.6: Save Project As Another File 18-10
18.3.7: Compress Project 18-11
18.4: Technology 18-11
18.4.1: Get the Number of Technology Layers 18-11
18.4.2: Get First Technology Layer 18-11
18.4.3: Get Next Technology Layer 18-11
18.4.4: Get Technology Layer Name 18-12
18.4.5: Set Technology Layer 18-12
18.4.6: Get Technology Layer Specification 18-13
18.4.7: Add Technology Layer 18-13
18.4.8: Delete Technology Layer 18-13
18.4.9: Get Technology Layer Handle 18-14
18.5: Cells 18-14
18.5.1: Set Current Cell 18-14
18.5.2: Get Current Cell 18-14
18.5.3: Get Cell Number 18-15
18.5.4: Get First Cell 18-15
18.5.5: Get Next Cell 18-15
18.5.6: Get Cell Name 18-16
18.5.7: Create New Cell 18-16
18.5.8: Create New Cell from Rectangular Clipping 18-16
18.5.9: Delete Cell 18-17
18.5.10: Rename Cell 18-17
18.5.11: Save Cell 18-17

18.5.12: Save As Cell 18-18
18.5.13: Save Selected As Cell 18-18
18.5.14: Discard Cell 18-18
18.5.15: Set Cell Read-only Flag 18-19
18.5.16: Get Cell Read-only Flag 18-19
18.5.17: Set Origin 18-19
18.5.18: Set Reference Point 18-20
18.5.19: Get Reference Point 18-20
18.5.20: Activate Reference Point 18-20
18.5.21: Get Window Rect 18-21
18.5.22: Is Ancestor of Cell 18-21
18.6: Layers 18-22
18.6.1: Set Current Layer 18-22
18.6.2: Get Current Layer 18-22
18.6.3: Get Layer Number 18-22
18.6.4: Get First Layer 18-23
18.6.5: Get Next Layer 18-23
18.6.6: Get Layer Name 18-23
18.6.7: Get Cell Layer Number 18-23
18.6.8: Get Cell First Layer 18-24
18.6.9: Get Cell Next Layer 18-24
18.6.10: Get Cell Layer Name 18-24
18.6.11: Generate Derived Layer 18-25
18.6.12: Generate All Derived Layers 18-25
18.6.13: Clean Cell Primitives in Derived Layers 18-25
18.6.14: Clean Cell Hierarchy in Derived Layers 18-26
18.6.15: Clean Derived Layers in the Whole Project 18-26
18.6.16: Clean Cell Primitives in Scratch Layers 18-26
18.6.17: Clean Cell Hierarchy in Scratch Layers 18-27
18.6.18: Clean the Whole Project in Scratch Layers 18-27
18.6.19: Delete Scratch Layers in Technology 18-27
18.7: Layout Objects 18-28
18.7.1: Get Object Specs 18-28
18.7.2: Add Object 18-28
18.7.3: Delete Object 18-28
18.7.4: Modify Object 18-29
18.7.5: Search 18-29
18.7.6: Find First Object 18-29
18.7.7: Find Next Object 18-30
18.7.8: Find Previous Object 18-30
18.7.9: Find Last Object 18-30
18.7.10: Get First Object 18-31
18.7.11: Get Next Object 18-31
18.7.12: Get First Cell Instance/Array Object 18-32
18.7.13: Get Next Cell Instance/Array Object 18-32
18.7.14: Box 18-32
18.7.15: Circle 18-33
18.7.16: Ellipse 18-33

18.7.17: Donut 18-34
18.7.18: Wire 18-34
18.7.19: Region 18-35
18.7.20: Text 18-35
18.7.21: Link Text 18-36
18.7.22: Unlink Text 18-36
18.7.23: Add Linked Text 18-37
18.7.24: Delete Linked Text 18-37
18.7.25: Delete All Linked Text 18-37
18.7.26: Get Linked Texts Number 18-38
18.7.27: Get Linked Text 18-38
18.7.28: Add User-Defined Property 18-39
18.7.29: Get User-Defined Property 18-39
18.7.30: Modify User-Defined Property 18-39
18.7.31: Delete User-Defined Property 18-40
18.7.32: Delete All User-Defined Properties 18-40
18.8: Tools 18-41
18.8.1: Merge Wires 18-41
18.8.2: Merge Selected 18-41
18.8.3: Resize Selected 18-41
18.8.4: Explode Selected 18-42
18.8.5: Explode All 18-42
18.8.6: Flatten 18-42
18.8.7: Cut by Line 18-43
18.8.8: Set Cut by Line Mode 18-43
18.8.9: Set Cut by Line Wire Mode 18-44
18.8.10: Get Cut by Line Mode 18-44
18.8.11: Wiring 18-44
18.9: Groups 18-45
18.9.1: Create Group 18-45
18.9.2: Remake Groups 18-45
18.9.3: Destroy Groups 18-45
18.9.4: Destroy All Groups 18-46
18.10: Selection Operations 18-46
18.10.1: Select Newest 18-46
18.10.2: Select Object 18-46
18.10.3: Select Layer 18-47
18.10.4: Select All 18-47
18.10.5: Deselect All 18-47
18.10.6: Invert Selection 18-48
18.10.7: Select by Type 18-48
18.10.8: Select by Box 18-48
18.11: Selection Modes 18-49
18.11.1: Set Select Mode 18-49
18.11.2: Get Select Mode 18-49
18.11.3: Set Concern Mode 18-49
18.11.4: Get Concern Mode 18-50
18.11.5: Set Fill Mode 18-50

18.11.6: Get Fill Mode 18-51
18.11.7: Set Ignore Instances 18-51
18.11.8: Get Ignore Instances 18-51
18.11.9: Set Ignore Primitives 18-51
18.11.10: Get Ignore Primitives 18-52
18.12: Edit Operations 18-52
18.12.1: Cut 18-52
18.12.2: Copy 18-52
18.12.3: Paste 18-53
18.12.4: Move 18-53
18.12.5: Duplicate 18-53
18.12.6: Rotate 18-54
18.12.7: Mirror Vertical 18-54
18.12.8: Mirror Horizontal 18-55
18.12.9: Move To Active Layer 18-55
18.12.10: Copy To Active Layer 18-55
18.12.11: Delete Selected Objects 18-56
18.12.12: Replace Selected Instances 18-56
18.12.13: Replace Instances in Cell 18-56
18.12.14: Replace Instances in Project 18-57
18.13: Support 18-57
18.13.1: File Name Selector 18-57
18.13.2: Update View 18-58
18.13.3: Refresh View 18-58
18.13.4: Refresh Object 18-58
18.13.5: Get Error Code 18-59
18.13.6: Get Error Message 18-59
18.14: Script Language (LISA) 18-59
18.14.1: XI Initialization 18-59
18.14.2: XI Run 18-60
18.15: User Interface (UI) 18-60
18.15.1: Get Mouse Point 18-60
18.15.2: Get Mouse Rectangle 18-60
18.15.3: Is Mouse Right Button Pressed 18-61
18.16: Library 18-61
18.16.1: Set Logical Name 18-61
18.16.2: Get Logical Name By File Name 18-62
18.16.3: Get File Name By Logical Name 18-62
18.16.4: Activate Library 18-62
18.16.5: Deactivate Library 18-63
18.16.6: Delete Library 18-63
18.16.7: Delete All Libraries 18-63
18.16.8: Get library Path Name 18-64
18.16.9: Get library Directory Name 18-64

Chapter 1: Introduction

1.1: About This Manual

EXPERT LAYOUT EDITOR and EXPERTVIEWS LAYOUT VIEWER are tools for designers of physical layout of integrated circuits. In this manual, both tools are referred to as EXPERT since both tools essentially share the same user interface. The only difference is that EXPERTVIEWS has no operations that change the layout of integrated circuits. Moreover, the user interfaces of GUARDIAN DRC and GUARDIAN LPE are basically the same as that of EXPERTVIEWS with the addition of specific verification operations.

Chapter 1: gives an overview of EXPERT'S features.

Chapter 2: introduces basic notions and terms used in EXPERT.

Chapter 3: gives a general description of EXPERT'S user interface.

The remaining chapters discuss EXPERT'S tools in detail.

Note: Those sections and paragraphs with features pertaining to EXPERT LAYOUT EDITOR only are marked by a † (dagger) symbol.

1.2: Overview of Expert Capabilities

EXPERT is the state-of-the art hierarchical full-custom design ULSI layout editor. It is designed to fit a wide variety of requirements for a modern IC design tool:

- no limitations on size of ULSI designs;
- intelligent geometry database, offering extremely fast navigation/processing speed and low requirements on computer memory;
- rich set of comprehensive tools for support of all-angle geometry designs;
- shared libraries system for distributed development environment;
- flexible tuning on different IC technologies;
- versatile customizable user-interface combining smart GUI with alphanumeric input;
- comprehensive, flexible setup;
- † integrated DRC with real-time (on-the-fly), on-line and batch modes of use;
- † powerful script language for command-line input, activity logs, autogeneration of layouts, and parameterized cells,
- † possibility of adding custom C/C ++ programs to control, edit, and modify EXPERT'S project data and user interface, using EXPERT API functions;
- efficient utilization of multithreading/multiprocessing;
- powerful plotting capabilities;
- convenient hierarchical navigation and selection.

The current version of EXPERT runs on PCs and workstations under Windows NT, Windows 2000, Windows XP, Solaris and Linux.

1.2.1: Advanced Capabilities

No Restrictions on Layout Complexity

EXPERT provides high-speed loading, editing and verification functions, and advanced graphical tools. It has no restrictions either on number of ULSI components, or on type of geometrical limitations set for specific layouts.

EXPERT'S original database allows efficient processing of ULSI without any restrictions on size, density, hierarchy complexity, number of cells, or geometry type. The dimensions of processed layouts are limited only by the amount of disk space.

† Script Language

A powerful script language in EXPERT consists of two main parts:

- **LISA (Language for Interfacing Simucad Applications):** LISA is a general-purpose language that supports all main program-flow controls, and provides the means for defining custom commands.
- **Xi-scripts** (EXPERT Interface Scripts)**:** Xi-scripts are an extension of the LISA by application-specific commands of EXPERT. They provide full control over layout objects and editing commands in EXPERT.

Xi-scripts provide the following capabilities in EXPERT:

- autogeneration of layout objects;
- creation of shapes of any complexity;
- command-line interface in EXPERT;
- custom editing commands;
- creation of parameterized cells;
- replayable activity logs;
- automation of routine activities: project verification, backup, etc.

User and Library Management for Teamwork

Shared Library projects provide robust and efficient concurrent access to common data by members of a design team.

Administrator's management of user access rights and library access types, set through file system permissions, ensures data integrity and security throughout the design cycle.

† Design Rule Checks

1. Highly efficient and flexible on-line and batch Design Rule Checker.
2. DRC in selected area.
3. Real-Time DRC, or DRC on the fly. Real-Time DRC watches while you are editing the design and reports violations immediately. It runs as a separate process in multithreading mode and takes full advantage of multi-CPU machines.

All modes of DRC impose no restrictions on type of geometry and complexity of layout.

Convenient visualization of errors is in both flat and hierarchical modes.

Project Tree

The chip navigation tool **Project Tree** presents the layout's hierarchy in the form of a level-by-level expanded hierarchy tree of instances/cells. It gives easy-to-use ways for exploring complex hierarchical VLSI designs. Project Tree allows fast loading of a required cell instanced at any level of the hierarchy, marking arbitrary instances on the background of the whole chip, and displaying selection and search

results. The option **Show all projects** shows the hierarchy for all libraries activated in current project. **Project Tree** is closely related with **Edit-in-Place**, selection operations, and **Cell Map**.

Hierarchy Manipulation

EXPERT delivers an advanced environment and tools to support the methodology of hierarchical design and editing of VLSI layout. These functions are provided by a number of specific Cell/Hierarchy operations, which make a user-defined logical subdivision of layout and full control of hierarchical structure possible. The key operations here are recursive Edit-in-Place, hierarchical view settings, flattening and grouping into cells.

†**Edit-in-Place** provides an easy and convenient way to edit objects of a cell within the surroundings of any its instance.

The group of **hierarchical view** operations creates a flexible environment for execution of basic editorial functions. These operations allow various levels of detail, ranging from showing the floorplan level, to presenting all-level objects together with instance frames.

Cell Map

EXPERT is equipped with an advanced layout geometry navigator for controlling opened layout windows. **Cell Map** is supplied with convenient tools for resizing/focusing of dependent windows. It allows you to view the layout map for current cell, and to globally monitor the corrections made in a locally viewed work area. If you click on a different cell name in the **Project Tree** window, **Cell Map** shows a cell preview for it. Double clicking on the cell name opens the cell.

Editing Tools and Design Accessories

†EXPERT supports polygon level entry as well as component level layout entry. A comprehensive set of objects is available for editing complex layouts (cell instances, arrays, polygonal regions, rectangular boxes, text for labeling of cells/primitives, multiple-style wires of any width, ellipses, and rolled regions). All-angles, 90° and 45° geometries, arcs and rolled drawings provide strong editing capabilities for high-frequency and mixed-signal designs.†

EXPERT is equipped with a flexible **grid setup**, providing both user-definable and automatic (zoom-adjustable) grids and subgrids in several representations.

There are convenient means for fast on-layout measurements (automatically graduated axes, flexible polyline rulers, distance meter and cursor locator).

Layer settings, arrangement, and visibility are easily controlled from the **Layer bar** and **Layer Setup** dialog.

Special Tools

1. Fast automatic **Extraction of Nodes** (as sets of electrically connected primitives from several layers).

† 2. **Clipping In/Out** of arbitrary layout areas with the capability of saving the obtained pieces as separate cells. The Clip-out feature preserves the hierarchy of instances that lie completely in the clipped area. The Clip-in function replaces the portion of the layout by the clipped-out cell.

† 3. **Resizing**, **rescaling**, and **fitting to grid** for process migration.

4. Integration with GUARDIAN DRC SYSTEM, GUARDIAN LPE NETLIST and PARASITIC EXTRACTOR and GUARDIAN LVS SYSTEM.

1.2.2: Summary of Features

Layout Design Options

- Full-custom design ASICs
- Any integration scale

- Any hierarchy depth
- Arbitrary-shaped geometry
- High-frequency circuits
- Any mask-defined technologies

User Interface

- Fast graphic display
- Customized workplace (design windows, menus, dialogs, toolbars)
- Customized design setup (grids, colors, layers, etc.)
- Customized shortcuts
- Various pan/zoom operations
- Custom menu
- Command-line interface

Hierarchy Manipulation

† • Convenient instance/array definition
† • Efficient global and selective flattening/exploding
† • Grouping selected objects into cells
† • Advanced recursive edit-in-place
† • Direct pick of objects anywhere in the hierarchy for edit-in-place
† • Replace instances
- Flexible means for presenting of instances/arrays in layout view
- Efficient Search of objects anywhere in design hierarchy

Project Tree

- Convenient design hierarchy navigator
- Coherent navigation through hierarchy tree and layout
- Global and local hierarchy trees for inspection of layout nesting structure

Layout Objects

- Wires of various styles
- Polygons (arbitrary, orthogonal, diagonal)
- Boxes, Squares
- Ellipses, Circles, Donuts
- Rolled regions (bounded by arcs and line segments)
- Cell instances and arrays
- Texts (labels) and polygonial texts

† Edit/Modify Operations

- Ergonomic interface with optional fine tuning via alphanumeric input
- Cut/Copy/Paste
- Rotate/Reflect
- Delete/Undelete
- Stretch (both angle- and segment- preserving)
- Add Jog Points
- Duplicate/Move
- Merge regions

- Subtract, create hole
- Cut by (poly)line
- Modification of properties
- Polygonize
- Resize/Scale

Import/Export

- Stream (GDSII) format
- CIF format
- Applicon format
- OASIS format
- LEF/DEF format

† Special Functions

- Fast built-in, batch, and real-time DRC
- Electrical node extraction and Node Probing
- Clipping out layout areas with hierarchy preserved
- Resizing and rescaling
- Offgrid checks and snapping to grid
- Netlist Driven Layout

1.2.3: Layout Database Handling

- Efficient access to external storage
- Real-time processing of huge layouts
- High performance support of layout manipulations
- Shared library system for teamwork

Selection Tools

- Flexible selection modes
- By mouse click/drag
- By rectangular box
- By object type
- Preselection from list of objects
- Selection of objects based on various criteria using Search functionality

Accessories

- User-defined grids/subgrids
- Automatic grids (adjustable-to-zoom ratio)
- Multiple grid representations
- Automatically graduated axes
- Flexible polyline rulers and distance meters
- Point/Line distance measurement
- Convenient toolbar for layer manipulations
- Powerful plotting capabilities

This page intentionally left blank.

Chapter 2:
Basic Notions

2.1: Projects, Cells, Libraries

The basic unit of layout design in EXPERT is a **cell**. It is a logical group of primitive layout objects (regions, wires, boxes, ellipses, labels) and instances of other cells. Although this group may be arbitrary, cells most often correspond to actual components of integrated circuits of varying complexity from transistors and logic gates to various functional blocks, such as adder circuits and memory blocks.

The **bounding box** of a cell, also called **cell domain**, is the minimal rectangle with sides parallel to coordinate axes that contain all objects of a cell. For an opened cell, the bounding box appears as a dashed frame box if you select **View→Cell View→Edited Cell Frame** from EXPERT'S menu.

Note: Text does not increase the bounding box of a cell. It is defined by geometric objects only. An exception is if a cell contains only text, then it defines the cell domain.

The placement of a cell within another cell is done by reference called **cell instance**. Such an operation is called **cell instancing**. When a cell is instanced in another one, the latter one is called **parent cell** for the former one, and the former one is called **child cell** for the latter one. It is also said that a child cell is **referenced** in its parent cell.

Using cell instances avoids multiple copying of the same geometric data when you use a cell several times. When we speak of instance **position**, we mean the position of the origin of the instanced cell within the parent cell.

Often, a cell is instanced within another one in a regular manner referred to as an **array instance**, or simply **array**. In an array, a cell is placed repeatedly in such a way that the origins of the cell placements form a rectangular grid with fixed offsets in X and Y directions called **array steps** or **array deltas in X and Y directions** respectively. Elements of an array are identified by the pairs of indices in this grid.

In the following, what is said about editing instances is also applicable to arrays, unless stated otherwise.

The set of cells containing layout data for a particular circuit design (such as the whole chip, a functional device, etc.) is called **project** or **library**. EXPERT stores projects in files on a disk. They are called project files or library files, which are in the same format. They are distinguished by typical usage. When you load a file in EXPERT directly, it is called a **work project**. If the main purpose of a file is to store cells that are used in other projects, then it is called a **library**. All cells of a project must follow the same design rules, and are supposed to be fabricated under the same technology.

EXPERT handles the following kinds of projects:

- A functionally complete set of cells. This means that if one cell is referenced in another cell (the latter cell contains an instance of the former one), then the former cell is present in the project. In other words, all cell objects are inserted into the project database.
- Projects that refer to cells in library files.

There are two types of **libraries**: exclusive and shared.

Cells from an **exclusive library** can be edited only by a single user at any moment. Other users can view and use cells from an exclusive library while being edited by someone else.

Cells from a **shared library** may be checked out for editing, and changes must be checked in to be visible to other designers.

2.1.1: Cell Name Restriction

Non-shared EXPERT databases (`*.eld`) are case sensitive in connection with cell naming, but shared (`*.sld`) is not case sensitive under the Windows platform. You can combine letters and numbers in any order. The following special characters are illegal for using in cell names: `*?"|:,;\/=<>()`. To comply with GDSII cell naming convention, and specifically with runningthe GUARDIAN DRC and HIPEX extraction tools from inside the EXPERT application, the following symbols are recommended to use in the cell names: `'1' - '9'`, `'a' - 'z'`, `'A' - 'Z'`, `'_'` and `'$'`.

2.2: Hierarchical IC Design

EXPERT makes it possible to decompose a complex circuit layout into smaller parts using the cell instancing approach. With EXPERT, all cells already included in a project can be edited and instanced in other cells, copied, or deleted. New cells can be created and imported from other projects or instanced from libraries.

Decomposition into cells simplifies the design of a chip. You do not need to correct the layout in all repeated parts because any changes done in a cell automatically affects all of its instances. On the other hand, hierarchical design is a way to reduce memory requirements. Only name and placement parameters are stored for each instance of a cell instead of the full list of cell objects.

The **top cell** (i.e., not instanced in any other cell) of a typical complete IC design is the whole chip. As a rule, it contains instances of complex cells. They in turn are made up of instances of smaller cells. The **leaf cells** (not containing instances) represent simple circuit elements.

To expose the structure of nesting cells within each other, use the hierarchy tree. The **Project Tree** window used in Figure 2-1 is the tree of parent-child relations for cell instances with **children** grouped by instanced cells.

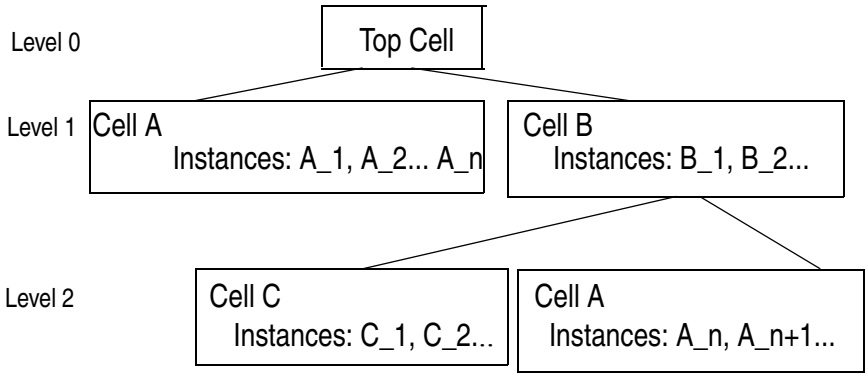


Figure 2-1: Cell/instance hierarchy outline

There are no restrictions on the number of cells used for a project, the number of instances of any cell at any level, or for the depth of hierarchy.

2.2.1: Dealing with Recursive Instancing

The only essential and natural constraint for hierarchical design is the following.

Recursive instancing is forbidden

This means that EXPERT will not allow you to put an instance of an ancestor of a cell into this cell, and a cell cannot be instanced inside itself. Cell B is called an **ancestor** of Cell C if an instance of B is present somewhere in the hierarchy tree of Cell C. In particular, this is the case if Cell B contains an instance of Cell C.

EXPERT enforces the rule against recursive instancing in two ways:

- All operations with cells are monitored, and those producing recursion are rejected.
- When a project is loaded or imported, the recursion check is performed. Suppose a cell `some_cell` is detected to be its own ancestor. Then, a non-existing cell named `some_cell_RECURSIVE` replaces an instance of `some_cell` in such a way the recursion is broken. After the loading/import is completed, the corresponding warning message is issued. This lists all these cells. You can then search for them and fix the problems.

2.2.2: Parameterized Cells

Parameterized cells, often called **P-cells,** add enormous flexibility to design and increase designer productivity. While the notion of cell allows you to avoid repetitive drawing of identical pieces of layout, P-cells go further: you can avoid repetitive creation of **similar** pieces of layout. As an additional benefit, you can quickly introduce changes into the layout by modifying the parameters of an instance of a P-cell rather than redrawing the geometry.

P-cells are created by writing a xi-script command in special format, compiling it and saving. After that, P-cells appear in cell lists of various commands. You can view them, place their instances, and change parameters of their instances through the Ordinary Cell Instancing Dialog.

See the description of commands **Cell→New** and **Edit→Create Object→Cell Instance/Array** for usage of parameterized cells.

See LISA: LANGUAGE FOR INTERFACING SIMUCAD APPLICATIONS USER'S MANUAL and Chapter 17: "Expert Interface Scripts (XI-Shell)" for syntax of the xi-scripts necessary for creation of P-cells.

2.3: IC Layout Design Concepts

2.3.1: Design Plane

A cell opened for editing in EXPERT is associated with the design plane endowed with the orthogonal coordinate system. The coordinate origin of the design plane is the cell's origin. In other words, the **cell origin** is at point (0, 0) in its design plane.

All objects of an opened cell are specified in terms of coordinates in the design plane. For wires and regions, coordinates are used to specify their vertices. For boxes, ellipses, and cell instances, coordinates are used to specify the position of their origin. Any mouse position on a visible part of the design plane is also associated with a pair of coordinates in the design plane.

Coordinates are computed by EXPERT'S internal functions. You can enter them either by the mouse, or from the keyboard using EXPERT input forms, including EXPERT'S command-line input.

When a cell is instanced, its origin is placed at the instance position, and all objects of the cell are placed preserving their positions relative to cell origin.

Coordinates are visible and accessible for you in terms of selectable measurement units. EXPERT gives you a choice of several common units and even allows you to define your own measurement unit. For internal computations, the **database unit** is used. Its value can be selected in **Setup→Editor/ Viewer...→General**.

Note: Since the coordinates are represented internally as integer values, there are limitations on the range of allowable coordinates. For example, if the database unit is equal to 0.1nm, this leads to the following restriction for machines with 32-bit integer arithmetic.

The maximum absolute value allowable for coordinates is 50,000 um, i.e., 5 cm. If this value is exceeded, the drawing will look strange due to "wrapping" integer numbers across their maximum.

Therefore, if you need to process larger layouts (e.g., **wafer-scale designs**) select the larger database unit in **Setup→Editor/Viewer...→General**. You can also use the **Use database unit from GDSII file** option of the GDSII I/O setup.

Note: In the current version of Expert, the work project and all libraries linked to it must have the same database unit. Otherwise, the result will be meaningless and EXPERT will issue warnings.

2.3.2: Layout Objects

An IC layout is designed in EXPERT in terms of **primitives**, which are actual shapes and **cell instances/arrays**. Additionally, **texts** are used for annotation purposes. Primitives and texts are said to be created in **layers**.

Layers

Layer is an attribute of any primitive geometrical object created with EXPERT. Some of the layers are of physical or technological meaning (for example, layers of polysilicon and metal), while other layers are conventional (created for verification or annotation purposes). Usually, it is said that an object **lies in a layer**. Layers have various user-defined attributes, such as name, color, GDSII number. See Chapter 11: "Setup", Section "Layer Color Setup" for details. EXPERT supports an unlimited number of layers. If you, however, plan to export your design using GDSII stream format, remember that many systems have restrictions on the allowable range of GDSII layer numbers and datatypes. The old GDSII specification sets the allowable range 0 – 63, and modern systems allow the range to be set to at least 0 – 255.

Primitives

The class of **primitive** objects in EXPERT is comprised of wires, regions, boxes, and ellipses. For convenience of shape creation, there are two special classes of regions: donuts and rolled regions. Often primitives are called **shapes**. We shall avoid the latter term in the manual because of a possible confusion: "the shape of a shape".

Primitives are characterized by the following:

- A primitive belongs entirely to a single layer;
- A primitive is not assembled from other layout objects.

When a cell is opened, its own primitives can be directly created, deleted or modified. Formal definitions of primitives are given below.

- **Region** is a part of the design plane bounded by a closed polygonal line (polyline) with at least three vertices.
- **Wire** is a parameterized region that may be tentatively defined as "thick line". Technically, a **wire** is constructed from a given polyline (named **centerline**) according to its parameters: **width** *(W)*, **joint style**, and **end style**, by the following procedure:

1. **W-Stick** is defined for every polyline segment to be a rectangle of width (W) and length (L) equal to the length of the segment. It is placed in such a way that the midpoints of the two of its opposite sides of length (L) fit the endpoints of the given segment (see Figure 2-2).
2. **E-Plaster** is generated for each endpoint of a wire to terminate it. **J-Plaster** is generated for each joint "to fill the cracks". Plasters are simple regions (triangles, quadrilaterals, pentagons, and disks) defined around the aforementioned points in accordance to wire **end and joint styles** as shown in Figure 2-2 and Figure 2-3.
3. **Wire** is created as a result of the union of W-sticks and E and J plasters. An example is shown in Figure 2-4.

In addition, in a 45° angle mode of drawing, the wire boundary is fit to the minimal grid. This is done by slightly increasing the width of diagonal sticks of a wire. Notice that for a non-90° geometry, it is impossible to have a constant-width wire fit exactly to grid. To cancel this grid fitting, set the **Off-grid wire boundary allowed** option in the **Setup→Technology→General...** menu.

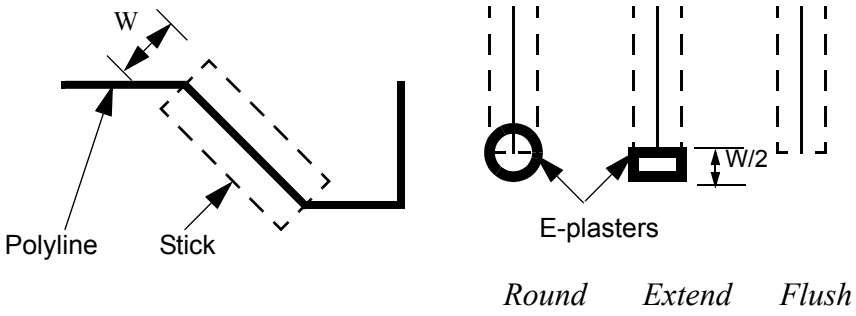


Figure 2-2: Sticks and End Styles

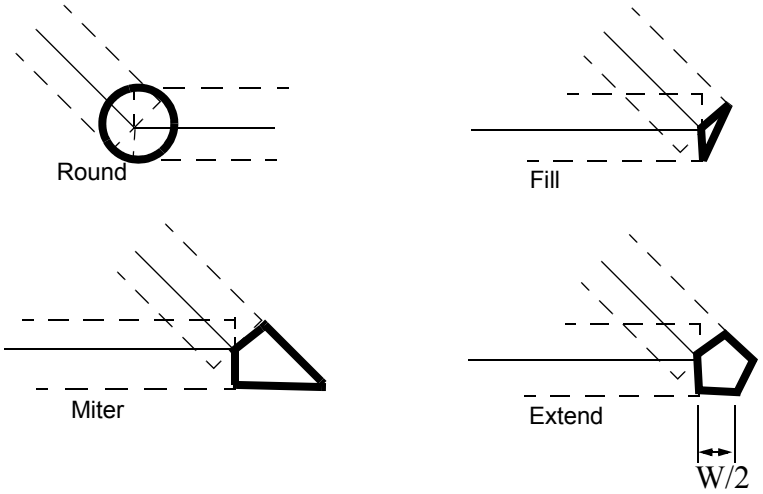


Figure 2-3: Joint Styles

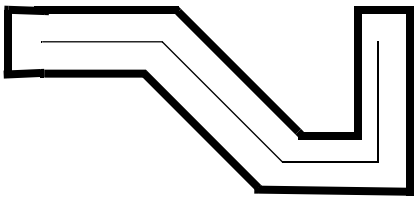


Figure 2-4: Wire with Extend ends and Miter joints

Note: The above description specifies only how wires will be shown on the screen. The actual shape of a fabricated wire depends on fabrication technology. For example, a miter joint style does not necessarily mean that two wires with touching "miters" will have electric contact.

- **Ellipse region**, or ellipse for short, is a part of the design plane bounded by a polygon that approximates ellipse with user-specified precision.
- **Circle** is a special case of ellipse.
- **Box** is an axes-oriented rectangle.
- **Donut** is a region of the design plane between two concentric circles.
- **Rolled region** is a region of the design plane bounded by a complex line, consisting of straight segments and arcs. For details in Chapter 5: "Creating Layout Objects", Section 5.4.5: "Rolled Region Creation".

Cell Instances and Arrays

Instance (of a cell) is a cell placed within another cell. An instance does not contain the actual content of the cell to be placed. It only **references** this cell and specifies the **position** of its origin and its **transformation** (if any). In the case of a parameterized cell (P-cell), it contains the **actual values of parameters**. This gives a considerable memory reduction in the case of multiple instancing of a cell. A cell may contain instances of other cells from the project as well as instances of library cells.

Array is a set of instances of the same cell with the same transformation positioned at the nodes of a regular grid in the design plane. The regular steps of this grid are called **row displacement** and **column displacement** (or alternatively, **row delta** and **column delta**) of the array.

We may naturally speak about rows and columns of arrays, and each array entry is associated with two indices conventionally called **row index** and **column index**.

Array position is defined to be the position of its (0,0) entry.

An array is specified by cell name, position, transformation, row and column deltas, numbers of rows and columns.

Text

Text (or label) is a string of alphanumeric symbols used for the documenting or labeling of objects. A text object has the following characteristics: layer, position, orientation, height, width, slant, and alignment with respect to its position. Optionaly, you can attach **text** to any non-text layout object.

Text is not fabricated. However, you can convert them to polygons from the Numeric Input Panel (see Chapter 5: "Creating Layout Objects", Section 5.5: "Text/Label Creation").

2.3.3: User Defined Properties

Objects may have user-defined properties. A property is specified by property name and property value. An object may only have one property with a given name. To export/import properties from/to `GDSII` file, use the property map file. You can select input and output property map files for user-defined properties.

This file consists of several lines. Each line has three items:

- `GDSII PROPATTR` value (positive integer)
- type (unused now)
- property name. It must be in quotes if it contains special characters or spaces.

Examples

```
9 all  node_name
```

```
12 all "Instance name"
```

Warning: If a `GDSII` file has some properties absent in the `"default.epm"` file, they are added with some default names.

Example

```
97 all "Property 97"
```

Selecting the menu command **Edit→Properties...** allows you to add/change attributes and user-defined properties of selected objects.

Selecting the menu command **Edit→Search...** allows you to search objects in the layout according to different search criteria.

2.4: Editor Accessories

EXPERT offers various precision editing features, visualization control tools, and setups for configuring system parameters.

- **Design measurement unit** can be chosen either from a predefined list or it can be assigned by you. See Chapter 11: “Setup”, Section 11.3: “Editor/Viewer Form” for details.
- **Grid** is produced by intersections of uniformly spaced horizontal and vertical lines on a design plane. Grid provides navigation in a coordinate system for placing objects.

There are several kinds of grids used in EXPERT:

- **Minimal Grid** defines the resolution allowable under the given technology. It is sometimes called “spot size”. Normally, all geometry produced by EXPERT lies on the minimal grid.
- **Snap Grid** is a grid for cursor movement (therefore for object creation) if snap cursor mode is switched on. The value of the snap grid step is set in EXPERT through Grid Setup.
- **Visual Grid** is the grid seen on screen. Its step is either user-defined or automatically adjusted so that some grid is always visible independently of the magnification of the layout. See Chapter 11: “Setup”, Section 11.4.6: “Grid Setup” for details.
- **Supergrid** is the grid with step multiple to the step of visual grid. It is useful for better size/distance reference.
  - Objects with any angles and sizes can be easily measured with using rulers, distance meter, and cursor position locator.
  - Optional graduated axes with cursor position marked on them and coordinate origins (see Chapter 11: “Setup”, Section 11.3: “Editor/Viewer Form” for details) makes the coordinate system visible.
  - Adjustable zoom and scroll steps (see Chapter 11: “Setup”, Section 11.3: “Editor/Viewer Form” for details) allow you to customize layout navigation.
- **Drawing styles** (choice between styles such as orthogonal, diagonal or arbitrary-angle modes, filling modes, and directional snap modes, etc.) can be changed at any time using the corresponding bars at the bottom of the screen.

Notice that any layer may have individual grid settings.

2.5: Files

EXPERT works with the following file types.

2.5.1: Expert Layout Database (*.eld file)

It is a binary file (with an `.eld` extension) containing the layout data, technology data, and system setup parameters in EXPERT’S internal format.

2.5.2: Shared Expert Layout Database (*.sld file)

Any opened `*.eld` project can be saved in an EXPERT shared format. It consists of a binary file with a `*.sld` extension and `*.cll` file folder containing `*.eld` files for each cell. This file format allows multiple users to access and edit a single EXPERT project using the Check-Out and Check-In functions.

2.5.3: GDSII file (*.gds file)

GDSII, or stream files, are binary files used to transfer layout designs between different CAD systems. Most often these files have the following extensions: `*.gds`, `*.sf`, `*.stm`, `*.str.` In some cases, however, they may have other extensions. For example, when DRC error reports are output in the stream format. To make EXPERT recognize these extensions, specify them in the GDSII I/O setup.

2.5.4: CIF file (*.cif file)

CIF files (with the default extension `.cif`) are ASCII files used to transfer layout designs between different CAD systems.

2.5.5: Applicon file (*.apl file)

Applicon files (with the default extension `.apl`) are binary files that store layout data in Applicon ("Apple") format.

2.5.6: OASIS file(*.oas)

Open Artwork System Interchange Standard (OASIS) is the new binary stream format created in response to the need for more efficient handling of growing data volumes, and is designed to replace the current GDSII format. OASIS is an efficient alternative to GDSII for post-tape-out data preparation flow processing. The OASIS stream file format reduces file size (3 - 10 times smaller the GDSII) for faster read/write, facilitates file transfer and data storage.

2.5.7: LEF/DEF file(*.lef, *.def)

A Library Exchange Format (LEF) file contains library information for a class of designs. Library data includes layer, via, placement site type, and macro cell definitions.

A Design Exchange Format (DEF) file contains the design-specific information of a circuit and is a representation of the design at any point during the layout process.

LEF defines the elements of an IC process technology and associated library of cell models. DEF defines the elements of an IC design relevant to physical layout, including the netlist and design constraints. LEF and DEF inputs are in ASCII form.

2.5.8: Technology files (*.tcn file)

The `.tcn` file extension defines layers and some design parameters of IC layout. It is used mainly when loading a project from foreign formats. Files in `.eld` format contain all necessary technology data. However, an external technology file can be used for `.eld` files by selecting **Setup→Technology→General...→Use external technology file for *.eld projects.**

In future releases, more parameters may be added to the technology file.

The technology file contains the following:

- **File format version number**

Modifying the `Version` field manually is not recommended. It indicates the version of the `FORMAT` of the technology file.

- **ID string** to identify a particular issue of the technology file.

The `TechnologyID` field of the technology file allows you to annotate the technology. For example, you can input the name of the technology and its revision number. It may be an arbitrary text with no length restrictions. The default value of this field is:

```
TechnologyID = "Non-named technology, ver. 0.0"
```

You can input or modify it by selecting **Setup→Technology→General...** menu, or by editing the technology file using a text editor.

- **Step of the minimal grid** (positive integer value, in nanometers).
- **Name of measurement unit**. EXPERT recognizes the following units: `nm`, `um`, `mm`, `cm`, `m`, `mil`, `inch`, `Lambda`, and the last name for a user-defined unit.
- **Unit value** in micrometers (if user-defined unit `Lambda` is used).

This field specifies the value of the user-defined measurement unit, `Lambda`. It gives the value of `Lambda` in nanometers. The default value for UnitValue is 100 nm.

Layer parameters for each layer consist of the following:

- **Layer name**
- **Layer color** definition: color name or RGB value, or both
- **Stipple pattern** name
- **GDS2 number**
- **GDS2 data type**
- **CIF name**
- **Wire parameters:** describes the shape of wires in the layer:
  - **Width:** is the default width value for newly created wires in the layer;
  - **Joint style:** describes the shape of wire segment joints;
  - **End style:** describes the shape of wire ends;
- **Derivation rule:** describes how the layer is to be combined from other layers. This rule is written in the DRC script language;
- **Individual Grid:** Any layer may have individual grid settings. If they are absent in the technology file, they are taken from the grid setup (**Setup→Technology→Grid**).

The following shows an example of layer definition.

```
GridParams
  {
  MinGridStep = 5000
  SnapCursorStepX = 1
  SnapCursorStepY = 1
  SnapDepend = 0
  SnapCursorRatioX = 0
  SnapCursorRatioY = 0
  GridColor = (255,255,0)
  GridType = 1
  AutomaticGridType = 0
  UserVisualGridStepX = 10
  UserVisualGridStepY = 10
  SuperGridStepX = 10
  SuperGridStepY = 10
  }
```

- **UseGridParams:** These parameters corresponds to the "**Individual grid**" switch in the **Setup→Technology→Layer/Wire Setup** menu. If its value is `TRUE`, the parameters from **GridParams** section will be used for the grid on this layer.

- **Layer connectivity:** Connectivity information is specified in the technology file as sets of layers connected by a connecting layer, or a via layer. The following shows an example of this format.

```
connect=("layer1" "vialayer")
connect=("layer1" "lay2" "lay3" "vialayer")
connect=("layer1", "lay2", "lay3";;;"vialayer")
```

The connectivity list may consist of any number of layer names. Layer names must be in quotation marks (""). The last layer is the connecting layer for the remaining ones. `connect=(  )` are obligatory elements of syntax. You may use any separators between layer names for readability.

- **Comment** is the part of a line between double slashes (//) and the end of line in the technology file.

Parameters not defined by a technology file are set to default values with the following exception. If layer color or stipple pattern or both are not defined, they are generated automatically.

- The **order** of layer definitions defines the order of their visibility on the screen with one exception: the **active** layer is always drawn over all other layers because new objects are created in the **active** layer and you want to be able to see them.

The technology file is in text format so you can easily edit them by any ASCII text editor. Names of variables are **case insensitive**. String values are **case sensitive**.

There are a number of other optional sections of the technology file supporting other Simucad CAD applications. They are described in detail in the corresponding manuals.

The example below shows a typical definition of technology parameters.

```
// Begin sample tcn file ================================================
Version= 100
TechnologyID= “Sample tcn file Ver. 1.10. Edited by nikK for UM 7/2/99“
Unit = um        // units of measurement is one of the follows:
                 // NM, MCM, UM, MM, CM, M, MIL, INCH, LAMBDA,
                 // nm, mcm, um, mm, cm, m, mil, inch, lambda, Lambda
UnitValue = 1.0  // size of user unit is actual if
                 //Unit = LAMBDA | lambda | Lambda
Grid = 100       // step of the minimal grid is 100 nanometers
Layer
  {
  Name    = "Metal2"  // name of the layer
  ColorName = "Blue"  // color name from library
  Color   = (0, 0, 255)  // color of the layer may be also defined directly
                    //as (R, G, B) value; BLUE is a predefined RGB constant
  Stipple = STIPPLE1  // name of stipple pattern
  GDS2Number = 22     // layer's GDS2 number
  GDS2DataType = 11   // GDS2 datatype
  CIFName = "CM2"     // layer's CIF name (string in quotation marks)
  Wire                // default  wire parameters for this layer
    {
    Width = 10         // width in current units
    Joint = EXTEND     // style of wire joints is one of
                       //EXTEND, FILL, ROUND, MITER
    End = FLUSH        // style of wire-ends is one of  EXTEND, FLUSH, ROUND
    }
  Derive = "AND: Layer1=Lr1,  Layer2=Lr2, LayerR=&Lr3;
          OR:  Layer1=&Lr3, Layer2=Lr4, LayerR=Layer27;" // Any DRC commands
  Scope = PRIMITIVES  // for derivation command is one of: PRIMITIVES,
                      //CELLWISE, FLAT
```

```
  }  // end of Metal2
Layer
  {
  Name    = "Layer28"
  Color   = (90, 90, 0)
  Stipple = STIPPLE2
  GDS2Number = 23
  GDS2DataType = 12
  CIFName = "TEMP"
  Wire
    {
    Width = 10
    Joint = MITER
    End = EXTEND
    }
  }  // end of "Layer28"
connect=(“Metal2” “Poly” “CM2P“)
// End of sample file ================================================
```

Note: Names in upper case represent names of predefined variables:

- `BLUE` is a shortcut for RGB color value;
- `NM`, `MCM`, `UM`, `MM`, `CM`, `M`, `MIL`, `INCH`, `LAMBDA`, `nm`, mcm, `um`, `mm`, `cm`, `m`, `mil`, `inch`, `lambda`, and `Lambda` are names of available units of measurement;
- `EXTEND`, `FILL`, `ROUND`, `MITER`, and `FLUSH` are names of predefined wire styles;
- `PRIMITIVES`, `CELLWISE`, and `FLAT` are names of predefined DRC scope modes.

Note: For Measurement Units in Technology File: Remember that in the technology file the (minimal) grid step (parameter `Grid`) is input in *nanometers*, but the wire width is defined in the measurement units specified in this technology file by the `Units` parameter. If you select the custom unit (`Lambda`), (`Units = “Lambda”`), then `Lambda` size must be in micrometers.

Example

```
Grid = 100 //in nm
Unit = "lambda"
UnitValue = 0.2 //in um
Layer{
  Name = "N-Well"
  Wire{
    Width = 1.0    // wire width equal to two minimal grid steps
    }
  }
```

In this case, the minimal grid step is 100`nm` and the default wire width in the `N_Well` layer is `1.0 * 0.2 = 0.2 um = 200nm`, or 2 grid steps.

Sometimes, it may be useful to set Lambda equal to the minimal grid step.

There are several ways to create a **technology file**:

- edit it manually in a text editor;
- create the necessary settings from EXPERT’S user interface (see Chapter 11: “Setup”), and then save them by selecting **Setup→Technology→Save Technology File...**;

- import the file from Dracula runset. See Chapter 11: "Setup", Section 11.2.1: "Technology Settings" for details. The latter possibility is useful when you are converting Dracula DRC decks.
- by selecting **Setup→Technology→Import Technology...** menu command.

Restrictions for Layer Names

Layer names cannot use the characters forbidden in file names. An additional list of restrictions is as follows:

- Layer name length must be at most 47 characters.
- The following characters are forbidden in layer names:
  - All non-printable characters (with codes 0-31 and above 126)
  - `= ( ) ; : " < > {}[], . + * \ / | ?`
  - The space character
- The first character cannot be '&'. It is reserved for marking temporary layers in the DRC script language.

2.5.9: Other Expert Files

Configuration files (*.ecf file)**:** This file stores data about the working configuration (the default) of the editor. See Chapter 11: "Setup" for description of settings stored in `.ecf` files.

Keyboard Shortcut Definition Files (`*.ekb`)**:** These files store keyboard shortcuts (hot keys) for EXPERT.

Color Definition Files (`*.clr`)

Stipple Pattern Definition Files (`*.stp`)

Plotting Style Files (`*.sty`)**:** These files control plotter presentation of layouts.

Layer List Files (`*.ell`)**:** These files are used for layer remapping during GDSII export and for storing layer visibility lists.

Expert Layer Plan Files (*.elp): These files store layer color and stipple pattern data, so layer style can be replaced without changing the technology file.

DRC Script Files (`*.dsf`)**:** These files contain design rule check command scripts.

Interface Script Files (`*.xis`)**:** These files contain xi-scripts (EXPERT interface scripts).

Some other file types used by EXPERT, such as log/message files, remapping files, and so forth are described in the corresponding chapters.

3.1: Expert Interface

This chapter gives a general description of EXPERT'S user interface. When starting EXPERT, an operating window, the pull-down menu, and a set of functional and information bars will appear on the screen. A sample layout of these elements is shown in Figure 3-1.

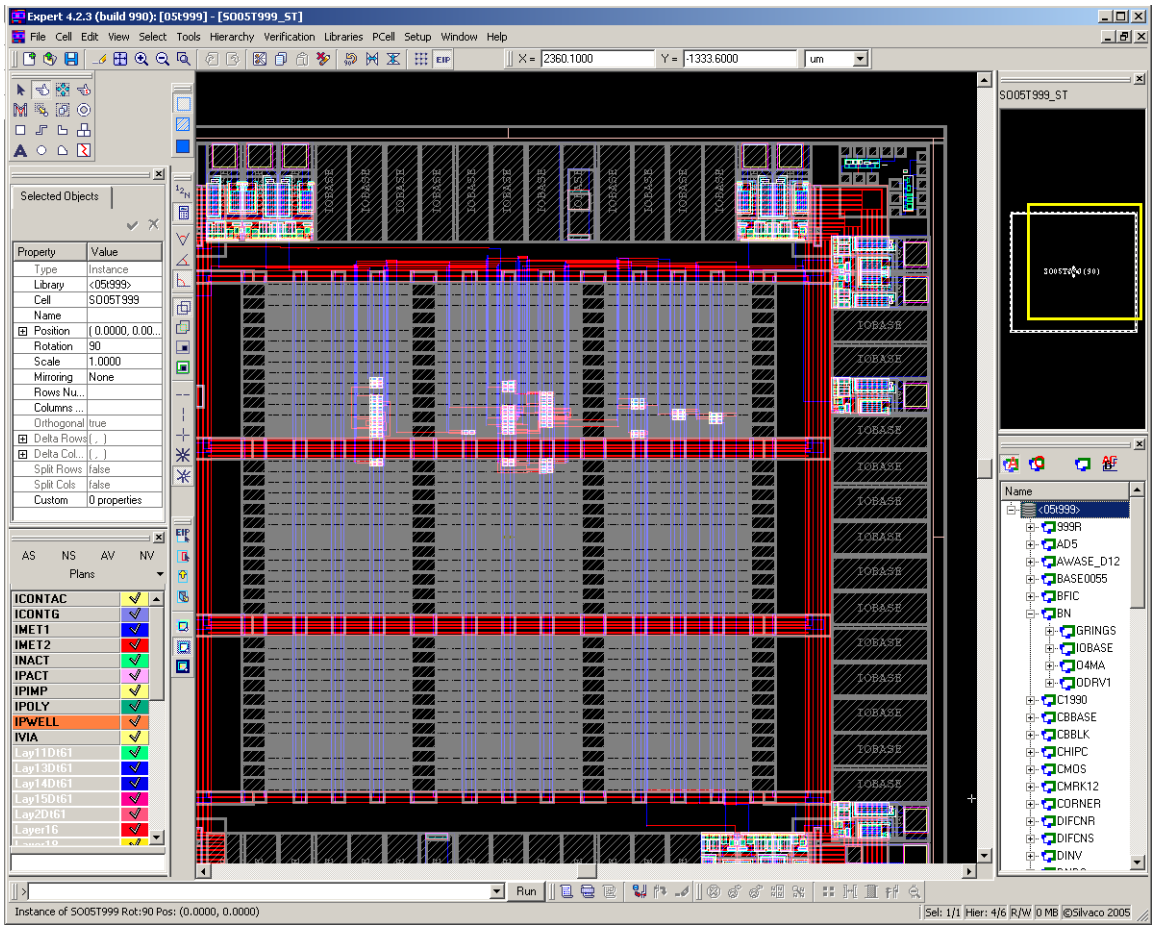


Figure 3-1: EXPERT's Interface: General View

Toolbar layout can be customized and stored between EXPERT sessions. It is also possible to remove all these bars from the screen (using menu command **View→Toolbars** or **View→Dock Windows**), and to run editing commands from the menu, from the command line, or by keyboard shortcuts. See Chapter 11: "Setup" on shortcut customization.

3.1.1: Menus

The Main menu (see Figure 3-2) controls all EXPERT operations. Clicking a menu item activates the corresponding operation or opens a submenu. Operations of close functionality are grouped into submenus. If a menu item is followed by ellipsis (...) then it invokes a dialog, for example, input of a command's parameters.

Figure 3-2: EXPERT's Main Menu

IC layout window shows the design plane for working with cell layouts. It is used for creating, editing and inspecting layout objects**.** You can open several cell layout windows. Layout windows can be tuned (i.e., magnified, shrunk, shifted, hidden, closed) either from the **Window** submenu, by the submenu **View→Dock Windows→Windows**, or by dragging their borders with the mouse.

For a larger layout, only a part of it is usually visible in a window. When you need to see other parts of the layout, or change magnification to see more, either use the options in the **View** submenu (in particular, **Zoom→Zoom-In**, **Zoom→Zoom-Out**, **Zoom→Zoom-to-Fit**, **Pan)**, the corresponding buttons from the **Command bar**, or the scrollers at the bottom and right borders of each layout window.

The simplest way of switching among layout windows is by clicking the left mouse button on the visible part of the window that you want activated. The required window can be chosen by name from the list in the **Windows** submenu. If the dock window **Windows** is open you can also directly double-click on the name of the window you want active. You can also press **Ctrl-F6** or **Ctrl-Shift-F6** to cycle through the open windows.

Most frequently used commands are placed into a number of toolbars or dock windows, which are described below.

Note: If you hold the cursor still over a button, a tool hint appears with a short description of the command as shown in Figure 3-4 (this can be disabled by the menu option **Display hints** in **Setup→Toolbars**). Full description always shown in status line..

The different **toolbars** available are: Align, Command, Design, Drawing, DRC, DRC error inspection, Real-Time DRC, Edit-In-Place, Filling, Hierarchy, Node Probing, Selection, Version control, NDL and LPE. A brief description of some of them is presented below.

- **Command** toolbar (Figure 3-3) contains a set of buttons for direct access to the most frequently used operations (each operation listed below corresponds to the numbered buttons in the figure). The contents of this bar may be different for different versions of EXPERT.

Figure 3-3: Command Toolbar

1. **File→New...**
2. **File→Open...**
3. **File→Save**
4. **View→Redraw**

5. **View→Zoom→Zoom to Fit**
6. **View→Zoom→Zoom In**
7. **View→Zoom→Zoom Out**
8. **View→Zoom→Zoom Into Window**
9. **Edit→Undo**
10. **Edit→Redo**
11. **Edit→Expert Clipboard→Cut**
12. **Edit→Expert Clipboard→Copy**
13. **Edit→Expert Clipboard→Paste**
14. **Edit→Delete**
15. **Edit→Rotate-90**
16. **Edit→Mirroring→Horizontal Mirroring**
17. **Edit→Mirroring→Vertical Mirroring**
18. **Setup→Technology...→Grid Setup...**
19. **View→Toolbars→Edit-In-Place**

- The **Design** toolbar (Figure 3-4) invokes tools for creating objects, instancing of cells, and typical selection and modification operations.

Figure 3-4: Design Toolbar

- The **Drawing** toolbar shows and assigns:
  - Angle restrictions for objects and for cursor movement
  - Modes for object creation (normal, merge, hole and hole selected)
- The **DRC** and **DRC error inspection** toolbars allow the user to invoke the DRC Script Panel, load the DRC errors, browse through the error, select the error marks type, and get detailed information on each error.
- The **Real-time DRC** toolbar (when enabled) displays the location of the error being flagged by the Real-Time DRC script.
- The **Filling** toolbar shows and assigns filling modes for drawing the regions.
- The **Hierarchy** toolbar allows you to select the view of the layout (lazy, flat) and the arrays display mode (full, perimeter, outlook).
- The **Node Probing** toolbar provides access to all node probing feature after an extraction has been performed.
- The **Selection** toolbar shows and assigns selection modes of object and the display mode of the current selected items.
- The **Version Control** toolbar allows you to Check in and Check out cells of a Shared Layout Data format (SLD) project. It also allows cell management function as New, Delete and Rename cell.
- The **NDL** toolbar allows you to show flight lines, highlight certain types of net, and show the pin flag when you are working with the Netlist Driven Layout (NDL) feature.

- The **LPE** toolbar allows you to specify the LPE setup, to invoke the desired parameter extraction, and to display the different netlist extracted (hierarchical, flat).
- All the toolbars can be customize for icon display mode and/or their content. **View→Toolbars→Customize...** invokes the **Preferences** window show in Figure 3-5. You can specify the size of the icons if a text label is present or not, or if a hint is displayed. The content of each toolbar can be modified by selecting it in the list and pressing **Customize...**.

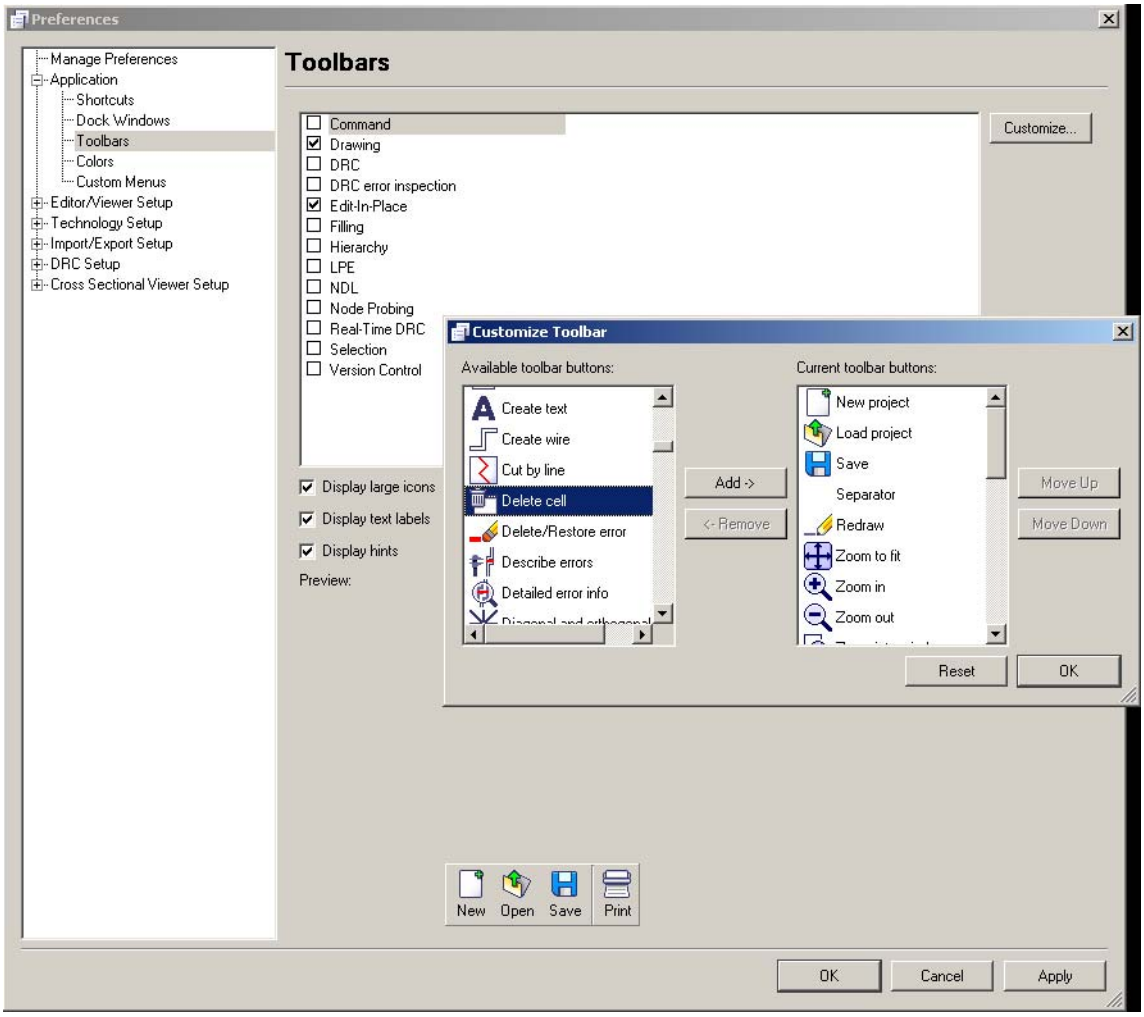


Figure 3-5: Customize

The different **Dock Windows** available are; Metric Bar, Net/Device Search Bar, Expert Log, XI-Command Line, Property Bar, Libraries, Project Tree, Objects Info, Layer Bar, Windows, Cell Map, Cross Sectional View, Cross Sections, Nets and Numeric Input. A brief description of some of them is presented below.

- **Metric Bar** dock window (Figure 3-6) contains the following items: cursor position and measurement unit.
- The **Net/Device Search Bar** dock window allows you to type in a net or device name and find it in the layout window.
- The **Expert Log** dock window displays all operations being performed during the EXPERT session.

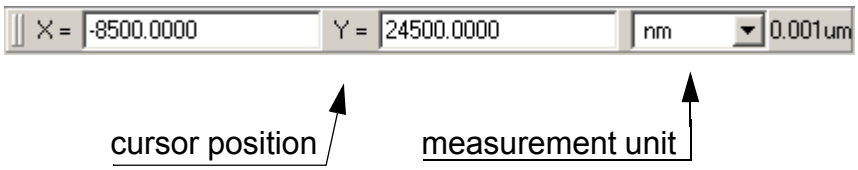


Figure 3-6: Metric Bar

- The **XI-Command line** dock window (Figure 3-7) inputs EXPERT'S interface script commands to perform editing operations.

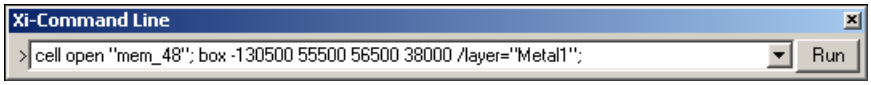


Figure 3-7: XI-Command Line Bar

- The **Property Bar** dock window (Figure 3-8) allows you to see and edit the properties of selected geometries.

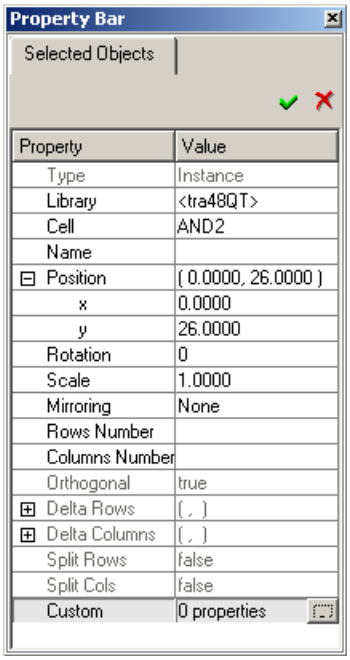


Figure 3-8: Property Bar

- The libraries status can easily be known by displaying the **Libraries** dock window. You can view and modify the library for a given project through this Libraries window. More specifically, it is possible to add and remove libraries, look at their properties, activate and deactivate a specific library for the current project and check in/check out cell. Chapter 15 will cover this section in more detail.
- The **Project Tree** dock window displays a view of the current project and its hierarchy. It can be used to navigate through the project. A double-click on one cell from the list will invoke the layout window of this cell. The icons at the top of the window allow you to view the **Project Tree** of the complete project, the active cell only, show the top level cell only, and display or hide the geometries.

- The **Objects Info** dock window lists the Name, Type and Location of the geometries under selection.
- The **Layer Bar** dock window (Figure 3-9) is used to specify the **active layer** (layer used to create objects), reordering of layers, make them visible/invisible, and so forth (see Section 11.2.1: "Technology Settings" for details about layer control).

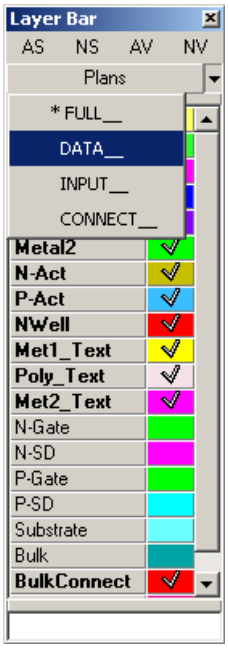


Figure 3-9: Layer Bar

- The **Windows** dock window shows the list of cell currently open. You can cycle through the open window by selecting a new one in the list. The organization of the window on the screen can also be modified through this interface window. The **Windows** dock window is shown in the top right corner of Figure 3-10.
- The **Cell Map** dock window displays at all time the location of your current view in the active layout window in relation to the full view of this same cell. A fat yellow outlined box is used to specify the location of the current view. The **Cell Map** dock window is shown in the bottom right corner of Figure 3-10.

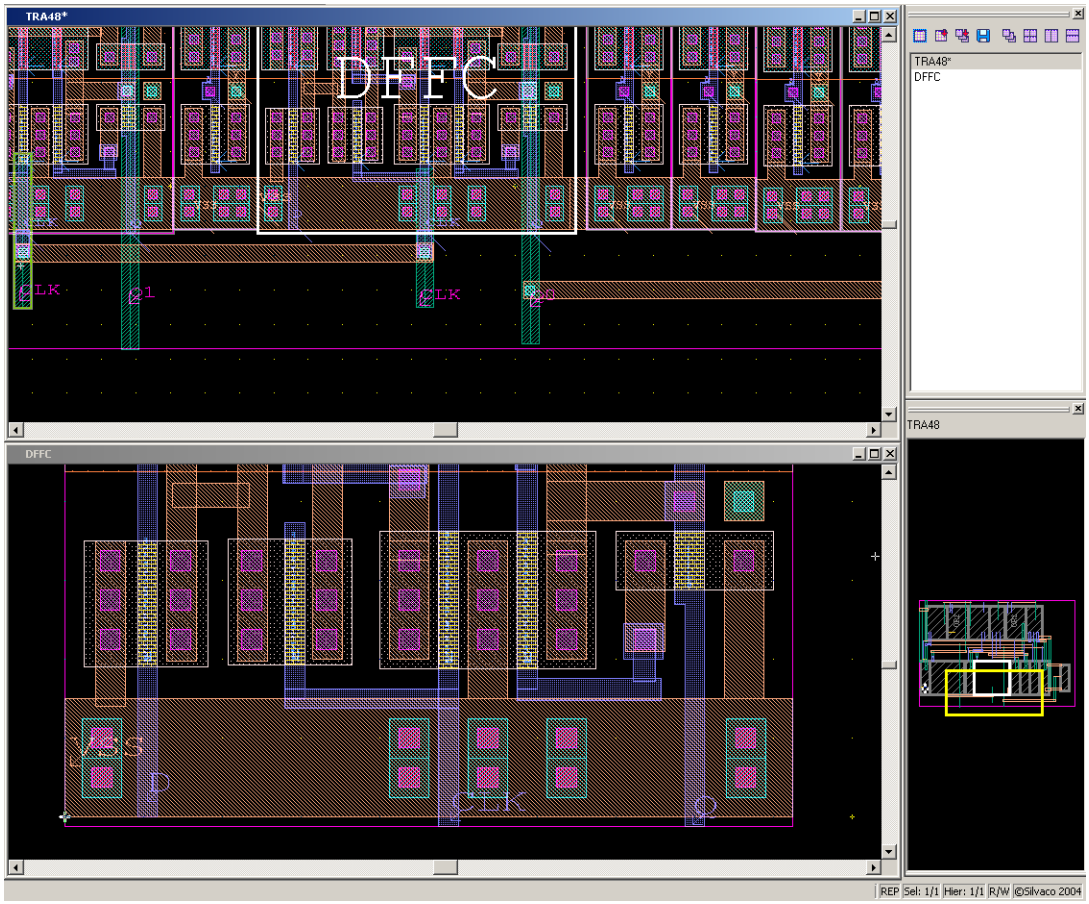


Figure 3-10: Window and Cell Map

- The **Nets** dock window allows you to display the flight lines for all nets or selected nets in different ways. This is mainly used when working with the NDL feature.
- The **Cross Sections** dock window allow you to input the coordinate of the required cross segment specifying where the cross section view needs to take place. This dock window is shown in Figure 3-11.
- The **Cross Sectional View** dock window displays the cross section of the process based on the specified cross segment. This dock window is shown at the bottom left corner of Figure 3-11.

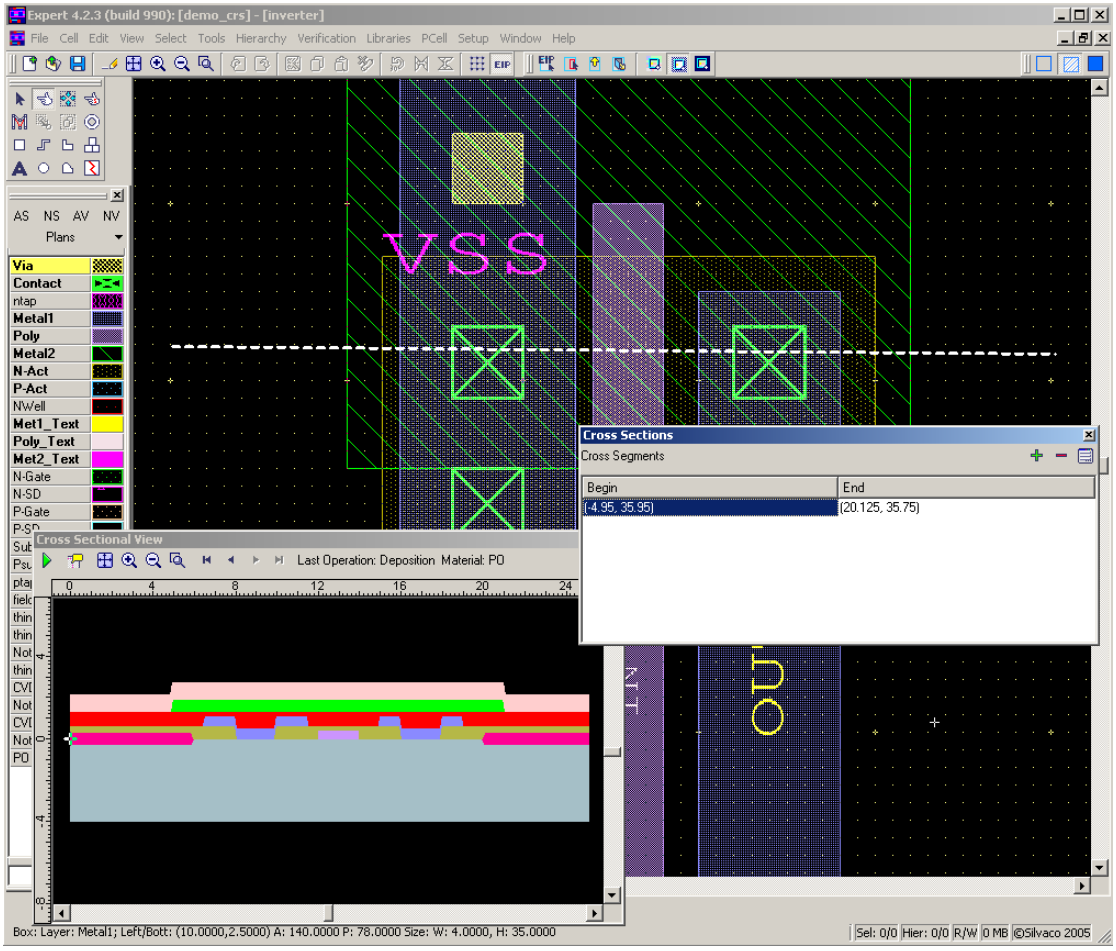


Figure 3-11: Cross Section Feature

- The **Window Title** on the top of the EXPERT window shows the project name, and the name of the cell opened in the active layout window. These names are shown in the following formats:

  `Expert:[PROJECT_NAME]-[CELL_NAME]` for ordinary work with an opened cell

and

`Expert:[PROJECT_NAME]-[CELL_NAME_0\...\CELL_NAME_N]`

for Edit-in-Place mode (see Section 10.3: "Edit-in-Place"), where `CELL_NAME_0, ...,CELL_NAME_N` is the ordered list of cell names that constitute the Edit-in-Place chain of nested cells.

A **Message Line** at the bottom of EXPERT'S screen provides messages and tips during an EXPERT session.

3.1.2: Toolbars and Dock Windows

EXPERT'S default screen layout with many toolbars or dock windows around is sometimes inconvenient: the user may desire more area for layout plane. You may switch them on/off by the menu command, **View→Toolbars** and **View→Dock Windows** respectively. It shows the list of check boxes corresponding to the different toolbars and dock windows. You may check/uncheck the specific item in this list and see the immediate changes in the screen layout. In both of these sub menus, there is the option **Customize** which, if selected, opens the setup panel allowing you to check which dock windows and toolbars you would like to be display. These two menu commands are also easily accessible by pressing the right mouse button when the cursor point is at the light gray area located at the edge of the EXPERT window.

Showing/hiding the most useful toolbars may be performed by keyboard shortcuts. They are assigned by the **Setup→Hotkeys/Menus...** menu command section **Application→Shortcut**. By selecting the Categories: View and the Commands: Toolbars→Command you can assign a shortcut, which can later be used to toggle the display of the toolbar Command. (see Chapter 11, “Setup” for shortcut customization)

The **Dock Windows** are resizable, both in height and in width. To resize the dock window you must do the following:

- position your cursor at the edge of the windows to resize and slowly move your cursor it until it changes from its current view to the a sliding cursor.
- press the left mouse button and drag left/right or up/down to the desired position then release the button.

All **Dock Windows** and **toolbars** can be undocked from the screen border and moved to a different position. To change their position you must do the following:

- position your cursor in the area free of controls (thin areas at the edge with a gray double line, see top of Figure 3-4), and wait for the cursor to change from its current view to a cross arrow cursor.
- double click the right mouse button to undock the window, or press the left mouse button, drag to the desired location and then release the button.

The **Dock Window** may also be resized while undocked.

Hint: In the default layout, the toolbars are docked to the screen border. You may undock the toolbar by one of the two ways described above and place it to another location. If you do not want a toolbar to be docked to any border of EXPERT'S screen, drag the toolbar until the outline of the dragged box becomes wider. If the bar was dock on the side of the screen, the direction of the outline box will also change from vertical to horizontal. By releasing the right mouse button the screen's border will be free of the docked bar, and the layout area will extended up to this border.

Hint: Sometimes you may wish to “sweep the toolbars under the carpet”, i.e., make them only partially visible, while the majority of a toolbar is being shifted beyond the computer screen border. Toolbar locations are saved/restored to/from ecf files and application registry, so that they are memorized between sessions of EXPERT. As a result, when EXPERT is loaded for the first time, the toolbar positions may be strange sometimes. You must place the toolbars properly and save the setup.

3.1.3: Mouse/Cursor Functions

Mouse Buttons

Most of the work with EXPERT is done via mouse. The mouse is used to **point to**, to **click on**, and **to drag**:

1. to operate on windows, dialogs, tool bars and menus;
2. to operate on layout objects inside the windows. The details of manipulations with the mouse in EXPERT are presented in their respective sections of the manual. This section only summarizes the terminology that is applicable for a description of your actions with the mouse.

- The mouse cursor is a small moving icon on the screen whose movement is controlled by the mouse device. The classical cursor's shape is an arrow. In EXPERT, the cursor's shape depends on which operation is activated as described in the corresponding sections. A cursor has a distinguished point called the pointer. It is used to precisely specify cursor position (or mouse position) on screen. The pointer is usually the arrow's tip, the center of a small cross, or some other clearly distinguished point of a cursor's shape.
- To **point:**
  Move the mouse to stop the cursor over the required object or screen area.
- To **click:**
  Press and release mouse button.
- To **double-click:**
  Click the mouse button twice quickly at the same point. For example in file selectors and cell selectors, double-clicking a file or cell name will open the file/cell under the cursor.
- To **drag:**
  Press and hold the mouse button, move the mouse in such a way that the cursor passes along the required path on the screen, then release the button. This action may be used, for example, to draw various boxes (for zoom area selection, shape creation, etc.).

Cursor Movements

There are several modes of movement of the cursor in the layout area:

- Smooth;
- Snap to Grid;
- Snap to Directions;
- Snap to Objects (Gravity).

Snap to Grid and **Smooth** modes are toggled by the **Edit→Snap Cursor to Grid** menu command.

In the grid snapping mode, the cursor may be positioned only at the nodes of the snap grid. This is valid for all editing tools (Measurement, Text, Modify, etc.), but not working for some others (Select by box, Zoom in/out window, etc.).

If the grid snapping mode is disabled, cursors such as Selection, Measurement and Text are allowed to be offgrid. However, all cursors to create or alter geometry shapes remain locked to the snap grid resolution specified in the grid setup. If offgrid geometries need to be placed, the snap grid resolution needs to be modified temporarily. Grid/snap grid parameters are set in the **Setup→Technology→Grid Setup...** (see Figure 3-12).

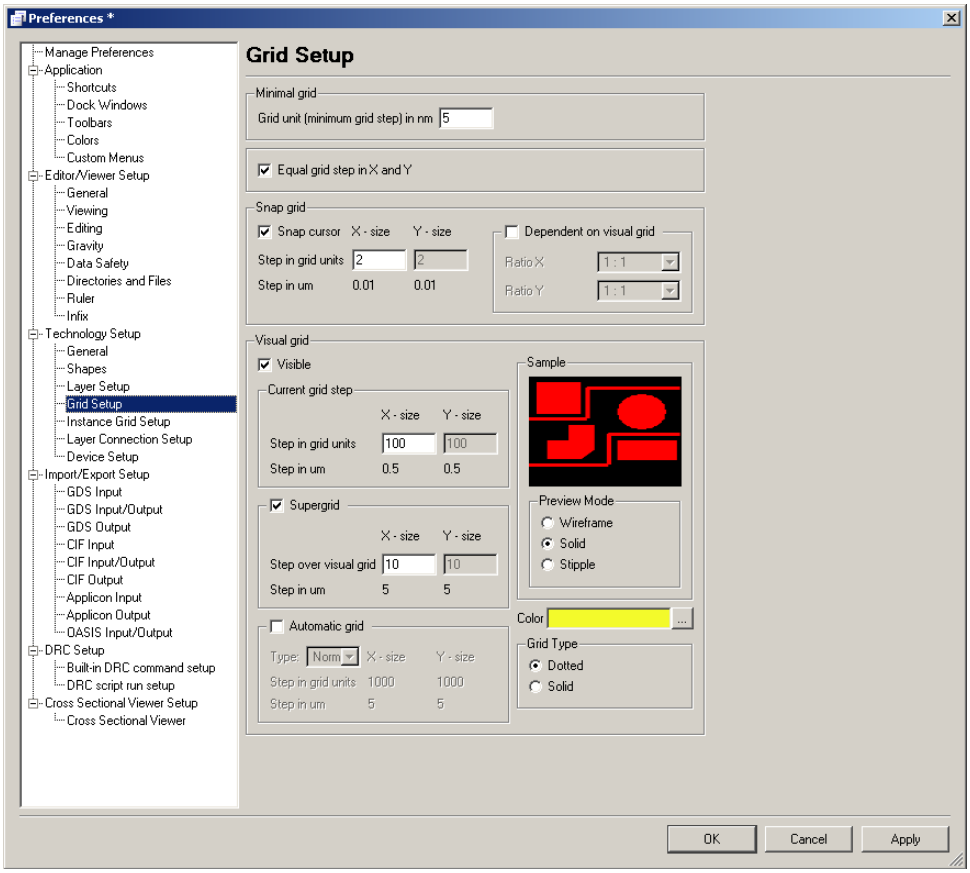


Figure 3-12: Grid Setup

In the setup of Figure 3-12 all geometries added or modified will snap to a step grid of 0.01um independently of the snap cursor box setting. Chapter 11: “Setup” will describe in more details the different aspect of the grid setup.

Directional Snap Modes restrict movements of the cursor to specific sets of directions during creation and modification of objects. To set the modes, either use **Edit→Direction Snap Modes** menu, or press the buttons at the **Drawing** toolbar.

Directional Snap Mode specifies how the cursor moves between two consecutive mouse clicks with respect to the point set by the first mouse click.

The following directional modes are available:

- None: cursor moves in all directions.
- Diagonal: cursor moves vertically, horizontally and diagonally.
- Orthogonal: cursor moves vertically and horizontally.
- Vertical: cursor moves in vertical direction only.
- Horizontal: cursor moves in horizontal direction only.

Gravity, or Snap to Objects, mode means that when the cursor comes close enough to shape elements (such as vertices and centers) it “jumps” to these elements. See Chapter 11: “Setup”, Section 11.3.8: “Gravity Page” for more information.

Bindkeys (shortcuts) for Mouse Functions

Sometimes, clicking mouse buttons may be inconvenient. For example, if a tablet is used instead of mouse, some buttons on the stylus (pen) may be inconvenient to press. To this end, the following are added: **Tools→Mouse Click→Left, Middle, Right**. If bindkeys are assigned to them, these bindkeys can be used instead of mouse clicks when drawing the shape and in some other operations. In addition, the **Enter** key acts as the right mouse click to terminate the drawing of polygonal shapes: region, wire, ruler and cutting line.

These replacements are especially useful if the mouse is not stable.

3.1.4: Dialogs

For many of EXPERT'S operations, you must input some parameters. Such input is usually performed from dialog windows (or simply dialogs) that appear on the screen when an interaction with the user is necessary.

When all required parameters are specified, as a rule the **OK** button is pressed to apply these parameters and to close the dialog (to force the program to accept parameters without exiting the dialog box, the **Apply** button is pressed, if it exists). The **Cancel** button is pressed to exit the dialog box without applying parameter changes.

The parameters may be input using the following dialog controls:

- **Edit boxes** are for parameters that you type in as arbitrary strings (for example, cell names). A special case is when only numerical values are acceptable. Such controls are referred to as **numerical fields.**
- **Drop-down lists** are used when text parameters have a predefined set of values. The **Drop-down** button (marked by a small black triangle pointing down), when pressed, displays the selection list. You can substitute the current value of a parameter for another one by clicking with the left mouse button on a value from the list.
- **Radio buttons** are convenient when a small list of mutually exclusive choices is available. To make a selection, click the mouse on the circle located near each value of the parameter (all of them are presented on the screen). The circle near the chosen item is filled. You can only switch on one circle from the radio button group at a time.
- **Check boxes** are used when there are a small set of **not** mutually exclusive choices. All these values are presented inside the dialog box accompanied by small boxes that are either empty (the corresponding choice/mode is switched OFF), or it contains the check sign (choice/mode is switched ON). Unlike the radio buttons, you can check several boxes (turned ON) simultaneously.
- **Tree control** is used to display data with a tree-like hierarchy. Most often this hierarchy is based on the hierarchy of cells in the project. This type of control is used in Project Tree (cell hierarchy navigator), in DRC error loading panel, and so forth.

3.2: Command Execution

Command execution can be started in various ways:

- from the main menu;
- from the custom menu (invoked by middle mouse button click in layout area);
- from toolbars;
- from the command line;
- by shortcuts;
- † by execution of DRC scripts;
- † by execution of xi-scripts, either from the xi-command line, or from the xi-script panel.

Execution of many commands is accompanied by **progress indicators** on the message line.

3.2.1: Breaking Execution

Many commands that take too long to execute (e.g., project loading) can be interrupted by hitting the **Esc** key. Following this action, EXPERT will then run the default tool, which is either the void tool or the selection tool, depending on the **Selection tool as default** setting in the **Setup→Editor/ Viewer...→Editing** form.

In addition, if a redrawing operation takes tool long, the **Redraw** icon changes to a **Red Stop Sign** icon which allows you to stop the redrawing process by simply pressing the new icon. It is useful when the layout is very large and you are drawing with many details.

3.2.2: Running Previous Command

Selecting **Tools→Previous Tool** menu command switches EXPERT into the previous editing tool.

3.3: Entering and Exiting Expert

To run EXPERT, you must select the command from Windows Start menu, or by shortcuts created during the installation of Simucad's CELEBRITY TOOLS.

Start→Programs→Simucad→Expert or double-click on the EXPERT icon on the Windows desktop panel to run the most recent installed version of EXPERT.

Start→Programs→Simucad→Versions→... allows you to run any previous version of EXPERT.

You may run several copies of an application on the same computer using one Simucad's license. To use this type of license, a special license file named "extended floating license" is required.

If using a Unix or Linux environment, EXPERT can be launch by typing `expert` in the following directory:

```
<installdir>/bin/
```

You can exit EXPERT in a number of ways.

- Hit **Alt-F4** on the keyboard while EXPERT'S window is active.
- On the Windows task bar, click right mouse button at EXPERT'S icon and select **Close** from the pop-up menu. Linux users can exit in a similar fashion.
- Choose EXPERT'S menu command, **File→Exit** from the pull-down menu.
- Press the X-mark at the top right corner of EXPERT'S main window.
- Not to say about killing EXPERT by the Application Task Manager.
- As a rule, the **Warning message box** appears on the screen (see Figure 3-13) after the exit actions listed above.

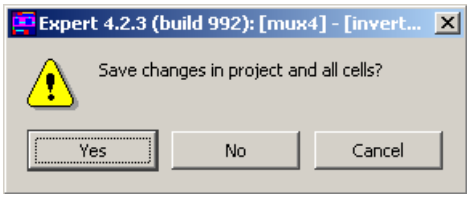


Figure 3-13: EXPERT Warning Box

Pressing the **Cancel** button returns you to the EXPERT session. If you press **NO**, the program exits and the latest changes are lost. If you press **YES**, EXPERT will save all changes made during the program session and exits.

3.4: Customization

3.4.1: Shortcuts

To assign keyboard shortcuts (also known as "hotkeys", "bindkeys") to any menu commands, select **Setup→Shortcuts→Setups...**. For more information about keyboard shortcuts, see Chapter 11: "Setup", Section 11.9: "Shortcut, Custom Menu, Custom Color Setups".

3.4.2: Color Setup

To change colors for various elements of EXPERT'S interface, such as selected objects, grid, and selection boxes, select **Setup→Setups...** and go in the section **Application** and **Colors**. For more information about colors, see Chapter 11: "Setup", Section 11.9: "Shortcut, Custom Menu, Custom Color Setups".

3.4.3: Custom Pop-up Menu

The custom menu will pop up if the middle mouse button inside the layout window is clicked. It contains a user-definable list of commands.

The most frequently used commands can be put into the custom menu by selecting **Setup→Setups...** and selecting **Custom Menus** from the **Application** section. There are three different custom pop-up menus: Select, Stretch and Other tools.

- Select: Allows you to specify the custom pop-up menu content when in selection mode.
- Stretch: Allows you to specify the custom pop-up menu content when in stretch mode.
- Other tools: Allows you to specify the custom pop-up menu content in all other modes.

Figure 3-14 illustrates the **Custom Menus** setup window, in which a user is in the process of adding "Ruler" to the Other tools Custom menu.

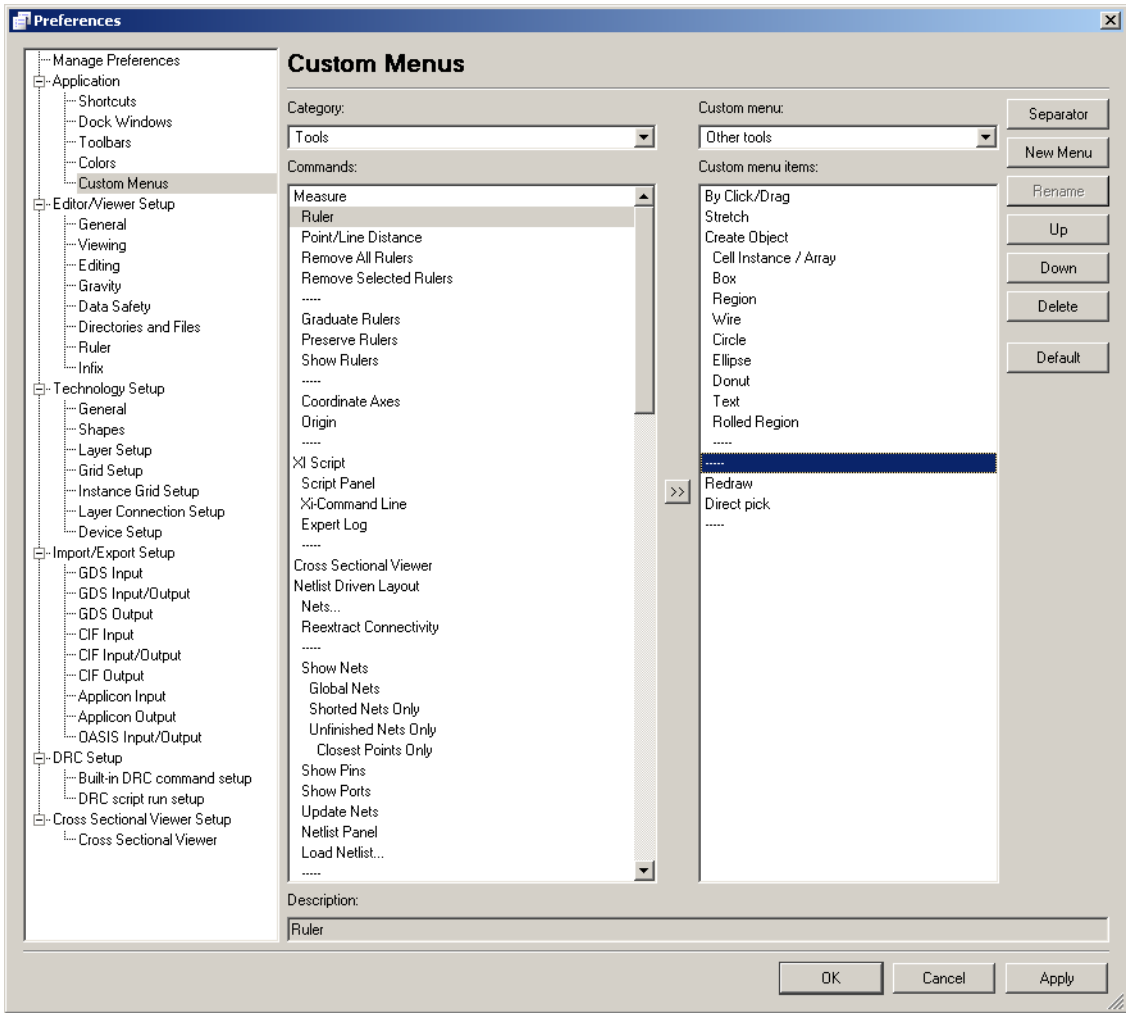


Figure 3-14: Custom Menu Setup

For more information about pop-up menus, see Chapter 11: “Setup”, Section 11.9: “Shortcut, Custom Menu, Custom Color Setups”.

3.4.4: Simple Custom Submenu

There is a simple way to add custom commands to EXPERT’s menu.

You can add menu items that execute xi-commands or procedures. For more information about these commands and procedures, see Chapter 17: “Expert Interface Scripts (XI-Shell)”.

To do this, define a file named `custom.mnu` with the syntax described below into EXPERT’s executable directory and start (or restart) EXPERT.

Below is an example of the `custom.mnu` file. To find this file, select the Simucad/Example/Expert/ folder of the CELEBRITY installation.

```
10 > &Debug
       &Box          = box 0 0 100 100; zoom;
       -----------
       Get &point    = get_point();
       > &Extra
         &Wordpad    = spawn("C:/WinNT/notepad.exe") /nowait;
```

```
            Wordpad &2  = spawn("C:/WinNT/notepad.exe");
            -----------
            &Box big    = box 0 0 1000 1000; zoom;
            > &Point
              Get &point = get_point();
              &Box3     = box 10 10 100 100; zoom;
            <
          <
          &Two Boxes    = box 0 0 100 100; box 50 50 100 100; zoom;
        <
11 > C&ustom
        &Box           = box 0 0 100 100; zoom;
        -----------
        Get &point    = get_point();
        <
12 > Test1
        &Pmos        = layer P-Act; box -2 -1 5 3; layer Poly; box -0.5 -2 2 5;
        Pmos&2       = box -2 -1 5 3/layer="P-Act"; box -0.5 -2 2 5 /
layer="Poly";
        Pmos&3       = box;
        <
```

Syntax

The syntax will be explained using the example above.

`10`, `11`, `12` are the positions in EXPERT's main menu for new submenus.

The > character marks the beginning of a submenu.

The < character marks the end of a submenu.

As you can see, submenus may be nested.

`&Debug` shows how to define the name (`Debug`) of a submenu and its hotkey (**Alt-D**).

`&Box = box 0 0 100 100; zoom;` line defines a submenu command. Here, as above, `&Box` defines the name and the hotkey for the command. It is separated from the actual command by the equation sign. Here, you can put several xi-commands separated by a semicolon, as in:

```
&Box3 = box 10 10 100 100; zoom;
```

The line:

```
Wordpad &2 = spawn("C:/WinNT/notepad.exe");
```

is an example of calling other Windows programs from EXPERT.

More advanced customization of EXPERT's User Interface and functionality is possible using Expert API (see Chapter 18: "Expert API").

Chapter 4:
Project/Cell Commands

4.1: Overview

The **File** submenu includes the following commands (see Figure 4-1):

†**New...:** Initializes a new project of ELD (Expert Layout Data format) or SLD (Shared Layout Data format) type.

Open...: Loads project from file in internal ELD, SLD or foriegn GDSII formats, CIF, Applicon, LEF, DEF or OASIS formats.

Close: Unloads current project.

Save: Writes the current state of the project to file on disk.

Save As...: Writes the current state of the project to new file on disk, possibly in a different format.

Save Cells...: Writes selected cells to file on disk.

Imports...: Import a project from a different format (GDSII, CIF, APL, LEF, DEF, OASIS).

Exports...: Export your current project to a different format (GDSII, CIF, APL, LEF, DEF, OASIS).

Reload: Reloads the current project from disk.

Info...: Displays various characteristics of the project.

Compress: Decreases the size the current project data file by performing **garbage collection**.

Page Setup...: Allows you to customize the printing operation to your specific requirements. You may be asked to select the printer to use.

Print Preview...: Displays a preview of the printing operation with the current **Page Setup** option.

Print...: Print a copy of the layout in accordance to the **Page Setup** option.

Print Status...: Displays information on the print job.

Recent Files: Displays the most recently opened files. This allows you to launch a previously opened file without having to go through **File→Open...**.

Clear all Recent Files: Deletes the list of Recent Files.. This option is located under the **Recent Files** option.

Exit: Terminates execution of EXPERT.

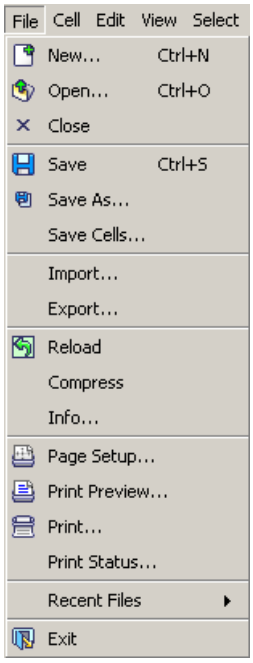


Figure 4-1: File Submenu

See Chapter 15: "Plotting" for more information about Printing commands.

The functions activated from the **Cell** submenu allow you to manipulate with whole cells (see Figure 4-2).

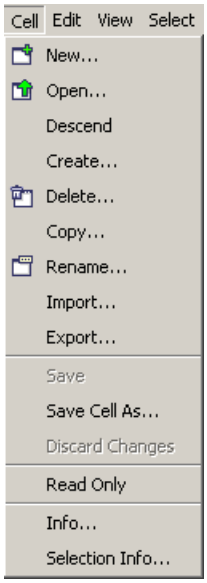


Figure 4-2: Cell Submenu

†**New...:** Opens a window allowing the creation of a new cell or parameterized cell (Pcell).

Open...: Opens an existing cell.

Descend: Opens the cell for the selected or preselected cell instance in another Layout window.

†**Create...:** Creates a new cell from the selected objects.

†**Delete...:** Deletes selected cells.

†**Copy...:** Duplicates cells (with new names).

†**Rename...:** Renames cells.

†**Import...:** Import a cell from one of many format (ELD, SLD, GDSII, CIF, APL, OASIS, LEF, DEF) in your current project.

†**Export...:** Export a cell from your current project to one of many format (ELD, SLD, GDSII, CIF, APL, OASIS, LEF, DEF, XIS).

†**Save:** Saves changes of the edited cell.

†**Save Cell As...:** Saves the edited cell under another name.

†**Discard Changes:** Discards changes in the edited cell.

†**Read Only:** Sets/Clears the read-only status of the active cell.

Info: Shows information about the cell and objects in it.

Selection Info...: Shows information about the selected objects in the active cell.

4.2: †Starting New Project

Purpose

To initialize a new IC layout design project.

Execution

- Select **File→New** from the menu or click **New project** button at the Command bar. You will see the **New Project** dialog window shown in Figure 4-3.
- Select the File type (file in EXPERT layout data format, with a `*.eld` extension or file in Shared layout data format, with a `.sld` extension).
- Select new project file name.
- Optionally, select a technology file as described in Section 4.2.1: “Technology File Selection”. If a technology was not selected, the project will start with the empty technology. You must then update it for your needs using commands from the **Setup→Technology** submenu. See Chapter 11: “Setup” for more information.
- Click the **Save** button.

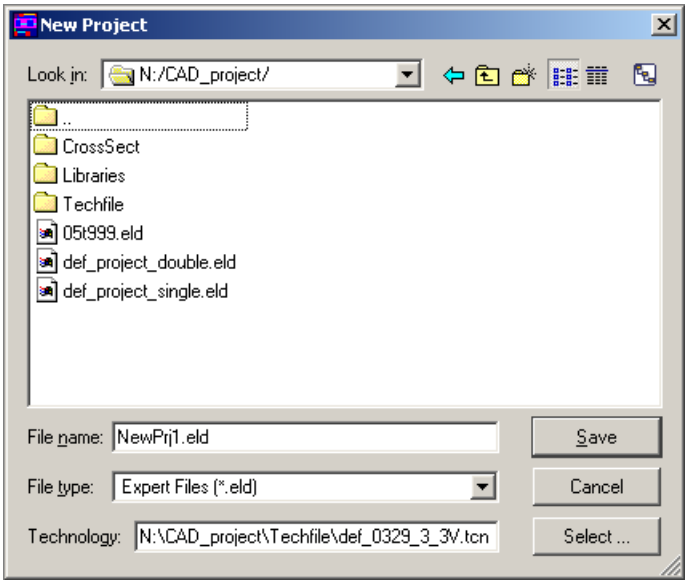


Figure 4-3: File Selector for New Project

Results

The **New Project** command starts a new EXPERT project with the selected file name and technology.

- If the option **Create new projects with default new cell** in the **Setup→Editor/Viewer...→Editing** panel is checked, then the Open Cell(s) window opens, showing you the new cell created. By pressing **OK**, the layout window opens, showing the empty design plane for this cell.
- Otherwise, the project will be created without a default cell. You will then have to use **Cell→New...** to create a new cell in the project.

Note: When a new `*.sld` project is created, the new cell created is in read write (R/W) mode by default.

4.2.1: Technology File Selection

Technology file selection is performed in the following cases:

- a new project is created;
- a project is imported from a file in foreign format (GDSII, CIF, Applicon, LEF, DEF, OASIS);
- EXPERT's project of type `*.eld` or `*.sld` is loaded with external technology file because the corresponding option is set in **Setup→Technology→General...**.

For all these operations, the **Load Project** window has a line Technology allowing you to specify the path of the technology file. You may either type it, or use the file browser button (labeled by **Select...**). If you click the **Select...** button, the file selector dialog will appear allowing you to browse for the required technology file. If the pathname of the technology file is shown, the file browser opens the directory of this technology file. Otherwise, the directory of the application default technology file is opened.

If no specified technology file is found, EXPERT prompts you to either repeat the selection, or proceed without technology. In the latter case, if you are importing/loading an existing IC layout, some default technology will be generated based on the layers present in the file. You may save this generated technology by selecting the menu command, **Setup→Technology...→Save Technology File...**.

If you successfully starts a project, the technology name is stored in the configuration settings, and saved between loads and sessions.

The specified technology file is loaded before opening the project. If a critical error occurs during the loading of the technology file, EXPERT returns back to the project file selector dialog. If noncritical errors occur during the reading of a technology file, the corresponding warning messages will appear.

Note: The **OS File dialog** button calls native file dialog for Operation System, and allows some additional actions such as Rename, Delete, and Set Properties for files.

4.3: Loading Project

A project can be loaded in many ways; the actual loading operation is described in the corresponding subsections of this section.

4.3.1: Open... Command

Purpose

Use **File→Open...** command to load a project stored in an ELD, SLD, CIF, Applicon, LEF, DEF, GDSII or OASIS file using a technology file to define layer attributes, if necessary.

Execution

There are some differences, whether you are loading a native (`eld` or `sld`) project, importing from a foreign format (GDSII, CIF, Applicon or OASIS), or loading a LEF, DEF format. They will be described in subsequent sections.

Selecting **File→Open** and the following extended file browsing form for project file selection will appear.

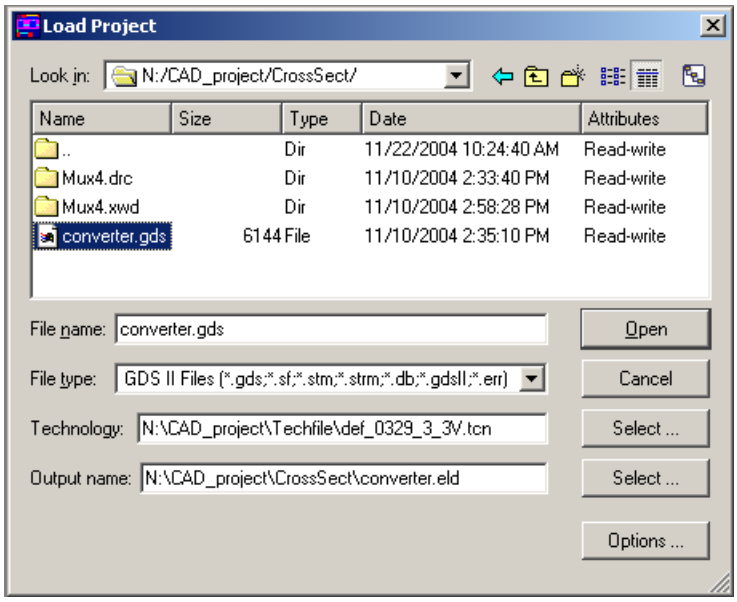


Figure 4-4: Project Selection Form

File type: You can choose ELD, SLD, CIF, Applicon, LEF, DEF, GDSII or OASIS file type for the project file to be loaded. Only files with the corresponding extension will be shown in the file list.

The allowable extensions for GDSII, OASIS, Applicon, and CIF files are listed in this field. They may be specified in the corresponding I/O setups available from the **Setup** submenu, or by pressing the **Option...** button. See Chapter 11: "Setup" for more information.

File Name: You must enter a file name in this field either by typing in, or using the standard file browser window.

Technology: You may define the path of the technology file as described in Section 4.2.1: "Technology File Selection". It is used when a project is loaded from a foreign format, or when **Setup→Technology→General...→General→Use external technology file for eld projects** is set.

Options: It allows you to specify certain option related to the type selected. For ELD and SLD type it will bring up the Data Safety setup, and when a project is loaded from a foreign format (GDSII, Applicon, CIF, or OASIS) it will open the corresponding setup option. See the next section and Chapter 11: "Setup" for more information.

As Read Only: You will see this option if you select eld type of files for loading. If this option is checked, the opened eld project cannot be modified.

Output name: This is the name of an ELD or SLD file. This is necessary when loading data from foreign format.

Backup name: You will see this field instead of **Output name** if an EXPERT project (`*.eld`) or (`*.sld`) is being loaded, and the option **Save original project upon loading** in **Setup→Editor/Viewer...→Data Safety** is checked. The **Select...** button to the right may be active or grayed out depending on the status of **As *.bak file** in **Setup→Editor/Viewer...→Data Safety**.

Cancel: Pressing this button abandons the **File→Open** operation.

Open: Pressing this button opens the selected file and a prompt to open a cell appears. If the project is successfully opened, the value of **File Type** field is saved in the configuration settings, and used as the default in subsequent project selections. However, you can override this behavior and select some fixed file type (e.g., `*.eld`) by setting the **Default type of project for load/save file** option in **Setup→Editor/Viewer...→General**.

Shared Access to Projects in Read-Only Mode

If a designer opens an exclusive library for editing, other designers can open this project in read-only mode.

4.3.2: Most Recent Used Project List

File submenu contains the **Recent Files** command, which opens a list of several most recently opened projects (MRU list). This list contains full path names of projects files. A mouse click at any line from this list loads the corresponding project.

If the corresponding project is in external format (GDSII, Applicon, CIF, etc.), then you are prompted to select a technology file and/or modify the I/O settings for the current external file format. See Section 4.3.5: "Loading from a Foreign Format" for more information.

4.3.3: Loading from the ELD format

If you press the **Open** button using the ELD format selected, a project with the file name `<File Name>` opens, where `<File Name>` is the name of the `ELD` file entered in the **File Name** field of the file selector dialog.

If the option **Use external technology file for `*.eld` format** in the **Setup→Technology...→General...** menu is selected, then the file selector dialog allows you to select the name of the technology file, as in the case of loading from foreign file formats. Otherwise, technology data stored in the `.eld` file will be applied.

After loading the file, the Cell Selection dialog described in Section 4.3.7: “Cell Selection Dialog” allows you to select a cell to be opened.

4.3.4: Loading from the SLD format

If the **Open** button is pressed using the SLD format selected, a project with the file name `<File Name>` opens, where `<File Name>` is the name of the `SLD` file entered in the **File Name** field of the file selector dialog.

If the option **Use external technology file for `*.eld` format** in the **Setup→Technology...→General...** menu is selected, then the file selector dialog allows you to select the name of the technology file, as in the case of loading from foreign file formats. Otherwise, technology data stored in the `.sld` file will be applied.

After opening the file, the Cell Selection dialog described in Section 4.3.7: “Cell Selection Dialog” allows you to select a cell to be opened.

4.3.5: Loading from a Foreign Format

When a foreign (GDSII, Applicon, CIF or OASIS) format is chosen for loading a project, technology selection is required because such files contain insufficient technology information. See Section 4.2.1: “Technology File Selection” for more information. `Eld` and `Sld` files store technology data inside themselves. In addition, because of format incompatibilities, you must specify certain options for translating data. The **Options** button at the file selector dialog box is used to invoke the setup dialogs for import from foreign formats, as is described in the corresponding sections of Chapter 11: “Setup”. For example, settings for import from GDSII format are described in Chapter 11: “Setup”, Section 11.5: “GDSII I/O Setup”. These options may also be preset by the commands from the **Setup** submenu corresponding to the foreign file type, for example, **Setup→GDSII Input/Output**.

4.3.6: Loading from LEF or DEF Format

When opening a file of LEF or DEF format, the technology selection is optional. If the **Technology:** field is left empty, the information within the LEF or DEF file will be used.

The **Output Name:** field is required. If no output name is specified, a file with the filename of the LEF or DEF file being open will be created with a `.eld` extension. You can specify an output name of type ELD or SLD.

If you choose the LEF format file, after loading the LEF file EXPERT will open and load the DEF file with same name (if exists).

4.3.7: Cell Selection Dialog

The Cell selection dialog contains the following fields.

Library List contains the name of the work project and the logical names of all libraries attached (linked) to this project. The name of the work project is shown in angle brackets. If you click on a name from the Library List, it is highlighted to indicate that it is selected. You may select one or several names from library list to see their cells in the Cell List.

Cell List contains names of cells (called ‘structures’ in **GDSII** format) present or visible in the file. If you click on a name from this list, it is highlighted to indicate that it is selected.

The following controls help in cell selection:

- The **Filter** field helps to list only the cells whose names obey a filtering criterion. A filter is a template for a cell name, which may contain two special symbols:? and *. The asterisk stands for any text, and the question mark stands for any single symbol. The asterisk is not displayed but implicitly present at the end of every filter.

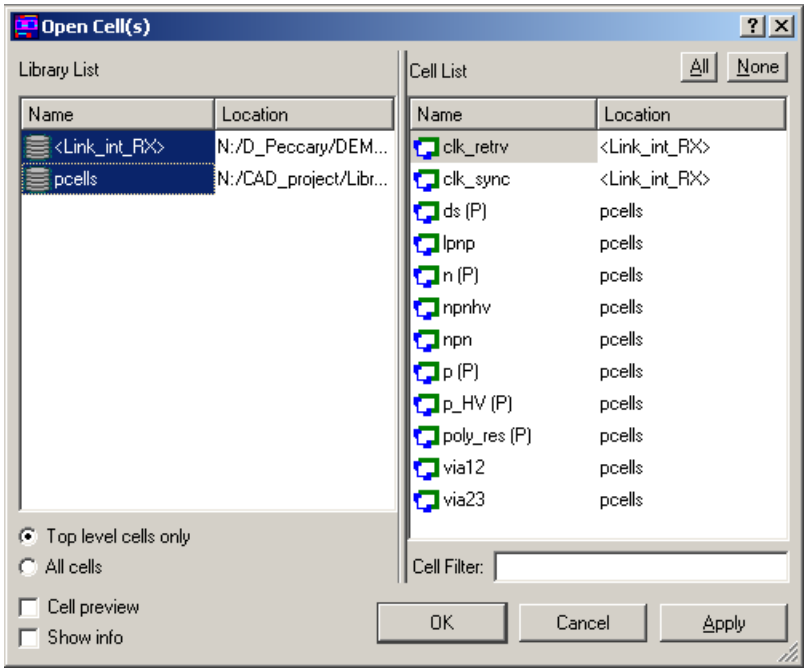


Figure 4-5: Cell open dialog box

Examples

- `*nv`: means that all cells with names containing `nv` will be listed: `env`, `inverter`, `nvid3`,...;
- `*p*`: means that cells with name containing letter `p` will be listed: `pie`, `ape`, `up`,...;
- `m?x`: means that all cell names with first `m` and third `x` will be shown: `mux1`, `mix`...;
- `?a*r*`: filters all names with the second letter `a` that contains `r` after it: `partran`, `layer1`.

To apply a filter, type it and press the **Tab** or **Enter** key.

To remove a filter to see all cells, clear the filter field and press the **Tab** or **Enter** key.

- **All Cells** radio button shows all cells from the Project file.
- **Top level cells only** radio button shows only cells that are not instanced in any other cell.
- **Cancel**: If this button is clicked, no layout windows will be opened after loading a file.
- **OK**: If this button is clicked, the selected cell will be opened in a layout window.

If you type the **Enter** key in a cell dialog, it behaves as follows:

- If the "filter" field is being edited (the caret cursor is in this field), the specified filter is applied and the corresponding cell name list is displayed,
- Otherwise, the selected cell is opened, as if the **OK** button has been clicked.

Every time **Open Cell** dialog is called, it uses a recently set up cell name filter and **All cells/Top level only** group choice that are preserved during the session.

Optionally, cell selection dialog may contain the cell preview window. You can turn it on by checking the **Cell preview** option.

You can also check the **Show Info** box to display more information about the selected cell. This information is presented in three subsections: Parent Cells, Child Cells and Layers. Figure 4-6 illustrates the "Cell Preview" and "Show info" feature.

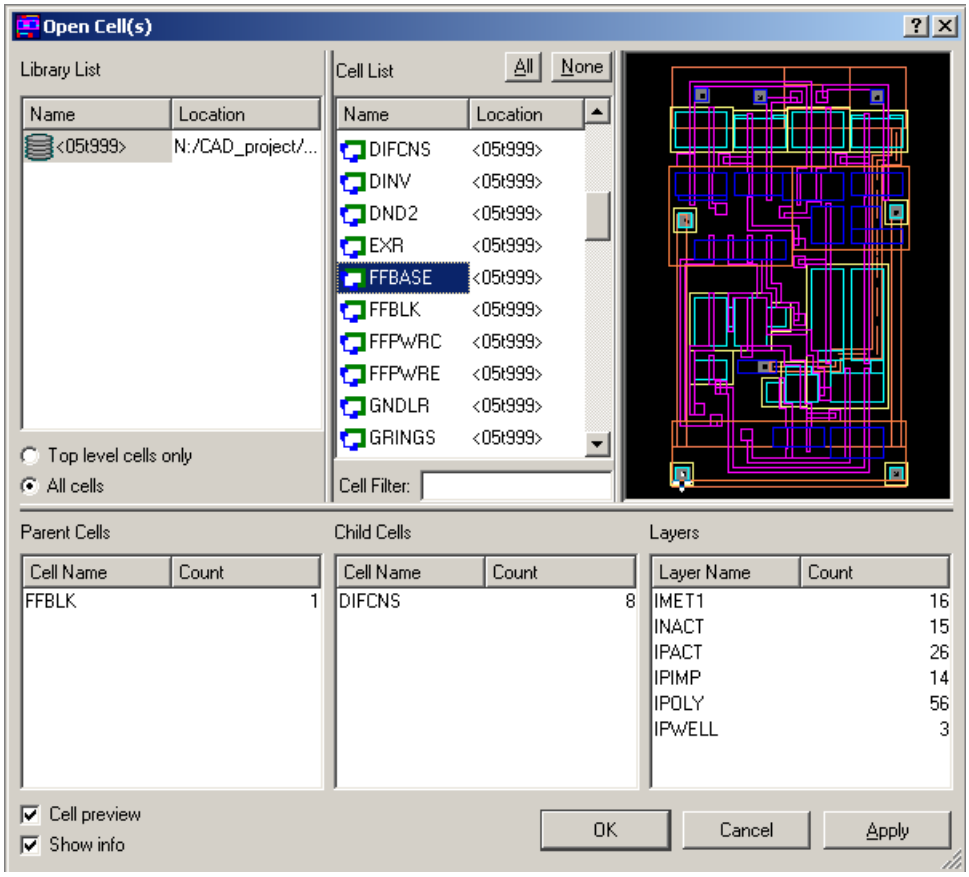


Figure 4-6: Open Cell(s) with preview

4.3.8: Possible Problems

Not Enough Space

When a project is loaded from a foreign format, EXPERT performs conversion into the eld format. Therefore, you will need sufficient space on the output disk.

The same caution is necessary when some backup options are turned on in **Setup→Editor/ Viewer...→Data Safety**.

Technology File Issues

File→New and **File→Open** (from a foreign format or with external technology file) may fail because of syntax errors in the technology file, or because some layers have the same (GDSII NUMBER, DATATYPE) pair. In particular, the latter may happen if the option **Allow layers with same GDSII numbers/datatypes** in **Setup→Technology...→General...** is not checked. Only layers with negative values of GDSII numbers/datatypes may have these values the same. (These values are not allowed by the GDSII stream data format specification, but may be used by EXPERT for scratch layers).

GDSII Format Issues

On Input

Two cases of wrong GDSII file format produced by some CAD systems are processed. The corresponding warnings are entered in the GDSII input log:

1. **One-point wires** (although forbidden by GDSII format) are converted into circles or squares, depending on the wire end style (rounded/extended), if the corresponding option in **Setup→GDS Input/Output...→GDS Input** is checked. One-point wires with flush end styles are always skipped.
2. A definition of **structure (cell) started within the definition of another cell**. In this case, the piece of the **GDSII** file between an ENDSTR record and the closest next BGNSTR record is skipped and some data may be lost.

On Output

1. When an attempt is made for writing a GDSII file with cell names longer than the number of symbol specified in **Cell name limit, symbol** using **Setup→GDS Input/Output...→GDS Output**, the cell name will automatically be modified to fit the symbol limit fixed. The last 7 symbols of the cell name will be given a unique instance index. Some CAD systems fail to read GDSII files with cell names longer than 32 characters.
2. When writing P-cells into GDSII file, new cells are constructed for individual P-cell instances. Cell names are constructed using P-cell name and parameter values for the P-cell instance in question.

   The Pcell name will be as follow:

   `<Library of the Pcell>_<Pcell name>_<list of parameter of the Pcell>`. This is only if the Pcell name does not exceed the number of symbol specified in **Cell name limit, symbol** using **Setup→GDS Input/Output...→GDS Output**.

   If it does exceed, cell names will be constructed from the P-cell name and a unique P-cell instance index (to conform to the number of symbol restriction for cell names). If there are several P-cell instances with the same set of parameters (disregarding usual parameters of instances, such as position and magnification), then only one "artificial" cell is generated.
3. You can also force the unique P-cell instance index approach by selecting **Encrypt pcell parameters** from **Setup→GDS Input/Output...→GDS Output**.

4.4: Saving/Compressing the Project

EXPERT has three commands for saving the projects: **Save**, **Save As...** and **Save Cells...**. In addition, the **Compress** command saves the project with "garbage collection" performed.

By these commands the edited project, linked libraries, or some cells from them are saved into ELD, SLD, CIF, Applicon, GDSII or OASIS format on disk.

4.4.1: Save Cells

When the **File→Save Cells...** command is executed, the cell selector dialog appears showing the list of all cells with non-saved changes. You must select the cells you want to save, and click the **OK** button. The selected cells will be saved on a disk in `.eld` format if the opened file is of `.eld` type, and in `.sld` format if the file is of `.sld` type. The remaining cells from this list will not be saved, but the changes will not be lost and may be saved later.

4.4.2: Save

When you apply **File→Save** command is applied to the `<PROJECT_NAME>` project, it saves the actual state of the current project to the `<PROJECT_NAME>.eld` file (rewriting it in the directory from where the file was opened). In addition, any changes in writable libraries linked to the current project will be saved as well.

When the opened file is of `.sld` type, The **File→Save** command will have no effect if all cells are in read only mode (R/O). If certain cells of the project are in read write mode (R/W or Check Out), the **File→Save** operation performs the equivalent of a Check In immediately followed be a Check Out operation for these cells. This is invisible to the user, but the state of the R/W cells is now saved in the project.

4.4.3: Save As

When the **File→Save As...** command is applied, the file selector dialog appears.

The extension chosen from the **File Type** field defines the format of the ELD, SLD, GDSII, OASIS, CIF, LEF/DEF or Applicon file. The name of the file where the project must be saved is typed in the **File Name** field of the dialog (an extension is added automatically), or a file name may be selected from the existing files (with the given extension) shown in the list of files. Click the **Save** button to save the project to file. Choose the **Cancel** button to abandon the save operation.

Result

The project is saved in the specified file.

If an existing file name is selected for saving, then a dialog appears that gives you a choice either to overwrite the file, or step back and select another file name.

Pressing the **Yes** button overwrites the existing file.

Pressing the **No** button returns you to the **Save As** dialog box.

4.4.4: Compress

The **File→Compress** decreases the size of the current project data file by performing "garbage collection". This operation can be useful after massive deletion of objects (e.g., whole layers and cells). After smaller deletions, EXPERT successfully utilizes free data file space for new objects.

The project is saved before compression.

4.5: Closing Project

Selecting **File→Close** closes the current project. In addition, **File→New...** and **File→Open...** attempt to close the previous project as well.

After these commands, EXPERT'S warning about unsaved changes may follow (see Figure 4-7). It means that there are unsaved changes in the work project, and/or in some linked libraries.

Yes: Pressing this button forces EXPERT to save the project (to the file `<P_Name>.eld` or `<P_Name>.sld`, where `P_Name` is the name of the project) before closing.

No: Pressing this button abandons the modifications made after the latest save operation and closes the project

Cancel: Pressing this button cancels the **Close** operation. You can save some or all changed cells by selecting **File→Save Cells...**, or you can discard changes since the last save/load in other cells by selecting **Cell→Discard Changes**.

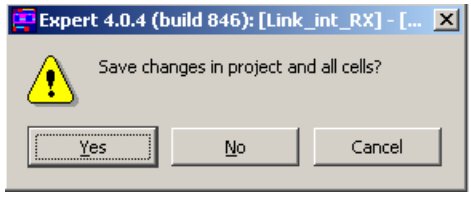


Figure 4-7: Expert Warning Box

4.6: †Cell->New

Purpose

Creates a new cell, either in the current project or in any attached library. Creating parameterized cells is described in Chapter 13: "Parameterized Cells".

Execution

Selecting **Cell→New** will open the **New Cell** dialog as shown in Figure 4-8.

Library List contains the name of the work project, together with the list of all activated libraries. The name of the work project is shown in angle brackets.

You have to select the target library and fill the **New cell name** field.

You can also click at an existing cell name in the **Cell List** to copy it into the **New cell name** field for editing.

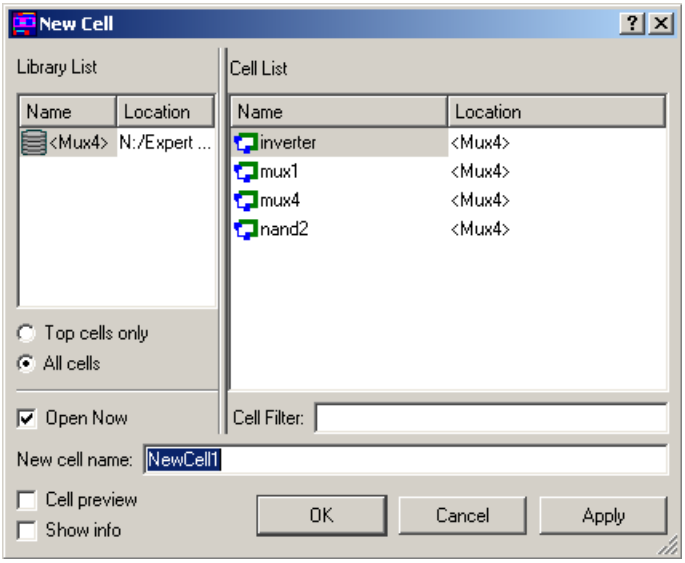


Figure 4-8: New Cell Dialogs

Results

Clicking **OK** will add the new cell to the project or library. A new window for the layout of the new cell is opened for editing if the check box **Open Now** is checked.

Note: You cannot create a new cell with the name existing in the target library.

4.6.1: Possible Problems

1. The library list does not show me the library to which I want to add a new cell.

Possible reasons:

- the library is not activated,
- the library is read-only.

4.7: Opening Cells

There are two commands to open a cell for editing/viewing in a **Cell** submenu:

Cell→Open... and **Cell→Descend**.

You can open several cells from exclusive libraries at the same time from the libraries panel (menu command **Libraries→Setup...**). See also **Hierarchy→Edit-in-Place** commands.

4.7.1: Open...

Purpose

Cell→Open... opens a cell from the currently processed project, or from libraries attached to it.

Execution

Click on the menu command **Cell→Open...** to see the Cell Selection dialog, which is described in Section 4.3.7: "Cell Selection Dialog". You must select a library from the library list, and a cell name from the cell list and click the **OK** or **Apply** button. You can have a Cell Preview or choose to display the cell information as described in Section 4.3.7.

Results

The layout window for the selected cell appears with the cell drawn to fit the window.

If you are opening a P-cell, the PCell Script will open automatically. You can close it and reopen it using the menu **Pcell→Show Xi Script**.

4.7.2: Descend

The **Cell→Descend** command is used to open the cell for a cell instance/array that you see in the layout window.

When the cell instance/array is selected, you can choose **Cell→Descend** to open the selected item in another window. This feature can best be used if a keyboard shortcut is assigned to it. When the item is selected, pressing the corresponding keyboard shortcut will descend in to the cell/array.

4.7.3: Ascend

The **Cell→Ascend** command opens the cell from which the **Descend** command was activated to open current cell. All cells are remembered so you can repeatedly make the **Descend** command and return back.

4.7.4: Possible Problems

1. The layout of the cell apparently does not fill the whole window.

The most probable reason is an erratic object away from the main layout. It can easily be detected as follows:

- Select the "normal part" of the layout by box drawn as tight to the "normal layout" as possible.
- Invert selection by the corresponding option from **Select** menu.
- See the number of selected objects by selecting **Cell→Info...** command.
- You might wish to look at these "anomalous" objects using **Edit→Search...** command, with the option **Search among selected**. If you check the box "Zoom to object", the layout window will display the selected objects.
- Delete the selected objects.

2. You cannot see the whole layout and the zoom-out function is disabled.

3. You see the cell together with some of its pieces duplicated.

The internal database unit is too small. You can change its value through the menu command **Setup→Editor/Viewer...→General**. This value must be changed before loading from a foreign format (GDSII, CIF, Applicon or OASIS) or after loading from an ELD or SLD file.

4.8: †Creating a New Cell from Selected Objects

Purpose

Creates a new cell from the set of selected layout objects. It is similar to selecting **Hierarchy→Create Cell in-Place** command.

Execution

Select either **Cell→Create...** or **Select→Save Selected→As Cell...** menu items. The **Create new cell from selection** box will appear with the following controls:

- **New Cell Name:** You must enter a name for the new cell into this field. Initially, this field contains a proposed default name (e.g.,`'Newcell1'`). It will be used if some other name is not entered.
- **Cell List:** This is the list of all cell names existing in the project. You can re-use these names to create new ones as follows. Click on a name from this list. It appears in the **New cell name** field where you can edit it.
- **Cancel:** Pressing this button abandons the opening of the new cell.
- **OK:** Pressing this button accepts the selected cell name to finish the operation.

If you press **OK**, there are two possible outcomes:

1. A message box **Cell "cellname" already exists** appears. This indicates that the project already has such cell name. You must press **OK** and choose another name for the cell.
2. **Create new cell from selection** dialog closes. The operation was performed successfully.

Results

If the operation was successful, the new cell is added to the project and its layout window opens. It is created from the selected objects of the active cell.

The coordinates of the objects in the new cell are obtained as follows.

1. The origin of the new cell is defined in terms of coordinates in the active cell as follows:
   - If a temporary reference point is active, then the origin of the created cell is set to this reference point.
   - In particular, you must set the reference point to (0,0), and activate it if you want the selected objects to have the same coordinates both in the old and new cell.
   - If no reference point is active, then the origin of the new cell is the lower left corner rounded down to the minimal grid.
2. The coordinates of the new cell origin are subtracted from the selected object's coordinates in the active cell.

Note: An instance of the new cell is NOT created in the active cell (select **Hierarchy→Create Cell in-Place** for this). The selected layout objects are NOT deleted from the active cell.

4.9: †Copying of Cells

Purpose

To create duplicates of selected cells in the target library or current project. Cell names are changed if necessary

Execution

Select **Cell→Copy...** and the **Cell Selection** dialog box will appears titled **Copy Cell(s)** (see Figure 4-9).

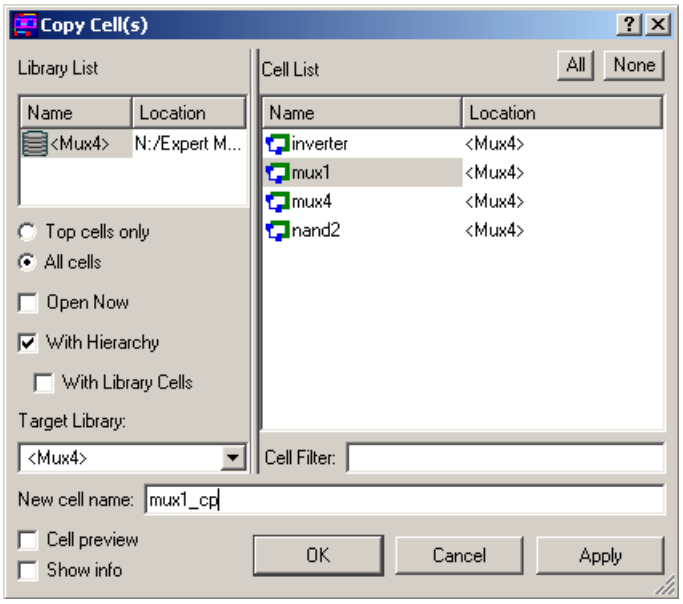


Figure 4-9: Copy Cells Dialog

Target Library: The library into which the cells will be copied. It may be a current project as well.

New Cell Name: If you select a single cell, you may assign a new name to the copied cell. If you select several cells, then EXPERT will try to copy the cells using the same names for duplicates. In the latter case, if cells with these names already exist in the target library, the Cell Name Collision Resolving Dialog will appear (see Section 4.9.1: “Cell Name Collision Resolving Dialog”).

With Hierarchy: If this option is set, then cells will be copied with all required subcells. If this option is not set, the references to subcell instances will point to the original libraries.

With Library Cells: If this option is set, then cells from the library will also be copied.

Open Now: If a single cell is selected for copying and this option is checked, then upon successful copying the resulting duplicate cell will be opened for editing.

Library List: From this list, you can select one or several libraries. To select several of them, hold down the **Ctrl** key while clicking on library names.

Cell List: From this list, you can select one or several cells. To select several cells, hold down the **Ctrl** key while clicking on cell names.

Filter: This field helps to list only the cells whose names obey a filtering criterion. A filter is a template for a cell name, which may contain two special symbols:? and *. The * stands for any text; the? stands for any single symbol. The * is not displayed but implicitly present at the end of every filter.

Examples

- `*nv`: means that all cells with names containing `nv` will be listed: `env`, `inverter`, `nvid3`,...;
- `*p*`: means that cells with name containing letter `p` will be listed: `pie`, `ape`, `up`,...;
- `m?x`: means that all cell names with first `m` and third `x` will be shown: `mux1`, `mix`...;
- `?a*r*`: filters all names with the second letter `a` that contains `r` after it: `partran`, `layer1`.

To apply a filter, type it and press the **Tab** or **Enter** key.

To remove a filter to see all cells, clear the filter field and press the **Tab** or **Enter** key.

- **All Cells** radio button shows all cells from the GDSII file.
- **Top cells only** radio button shows only cells that are not instanced in any other cell.
- **Cancel:** If this button is checked, the **Cell→Copy** operation is abandoned.
- **OK** or **Apply:** Pressing any of these buttons completes the copy operation. Upon completion, **OK** closes the copy dialog while **Apply** leaves it open and refreshes cell lists for more copy operations.

If you type the **Enter** key in a cell dialog, it behaves as follows:

- If the **Filter** field is applied (the caret cursor is in this field), the specified filter is applied and the corresponding cell name list is displayed,
- The selected cell is copied as if the **OK** button is clicked.

Every time Copy Cell Dialog is called, it uses recently set up cell name filter and **All cells/Top cells only** group choice that are preserved during the session.

If you press **OK** or **Apply**, there are possible two outcomes:

1. The operation will perform successfully.
2. The **Cell Name Collision** dialog box appears as described in the following subsection. The cells will be copied after resolving all collisions.

Note: To copy the currently edited cell, you can select **Cell→Save as**.

4.9.1: Cell Name Collision Resolving Dialog

This dialog is shown in Figure 4-10. The following operations are provided for resolving pairs of colliding cells:

- Select cells by clicking the individual names, or **All Copied** or **All Existing** buttons.
- Replace selected old cells by the new ones.
- Skip selected new cells.

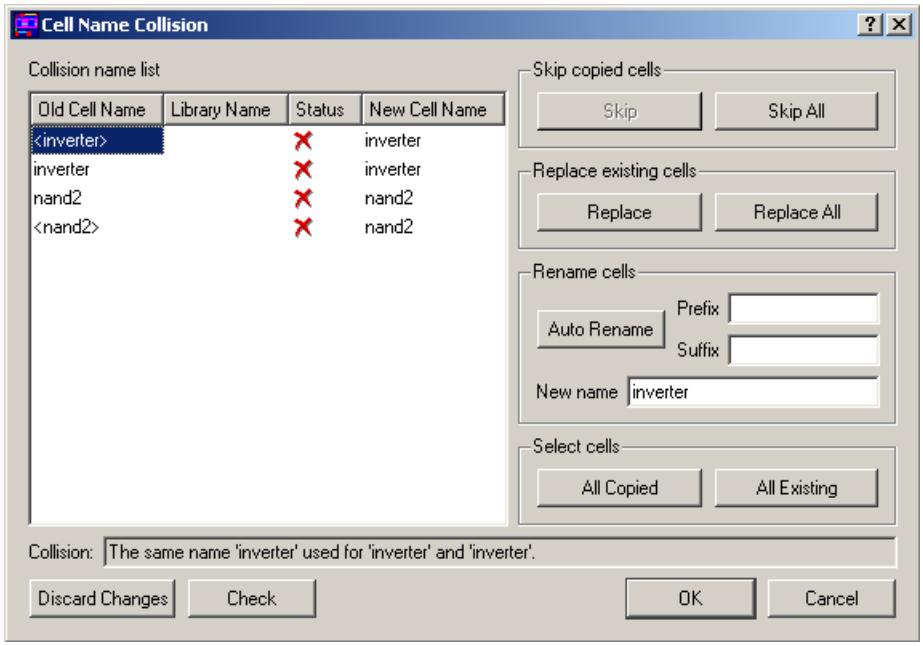


Figure 4-10: Cell Name Collision Resolving Form

- Rename selected cells by appending a user-defined suffix or prefix. To do this, type the suffix or prefix or both and click on **Autorename** button.
- Rename a selected cell by a typing the name in the **New name** field. It will immediately appear in the **New Cell Name** column.
- Clicking the title of a column sorts the collision list in ascending or descending order (by old or new name, or by library name).

If you get lost during complex collision renaming, click the **Discard Changes** button to restore the initial state of the dialog.

4.10: †Deleting Cells

Purpose

Delete cell(s) from the project/library.

Execution

Click onto the **Cell→Delete...** menu item. The **Cell Name Selection** box appears with the **Delete cell(s)** title.

The **Library List** contains the list of all exclusive libraries linked to the work project, including the current project.

You can select one cell or several cells to delete.

If a cell to be deleted is instanced in other cells, a warning box appears stating that these instances will also be deleted.

With Hierarchy option:

If this option is off, and a cell to be deleted has instances of other cells, the latter ones are not deleted, even if they are no longer instanced anywhere else (they become top-level cells of the project). Otherwise, the selected cells that belong to the work project are deleted together with cells from this project that are no longer instanced anywhere else.

If the cell to be deleted is currently active in the layout window, **Cell→Delete...** will open the **Cell Name Selection** box and the active cell will be highlighted.

4.11: †Renaming of Cells

Purpose

Renames a cell.

Execution

Select **Cell→Rename...** menu item and the **Cell Name Selection** box titled **Rename Cell(s)** will appear.

The following fields appear:

- **Cell List:** This is the list of all cell names existing in the project. Clicking on a name from this list selects a cell to rename, and copies its old name into the **New cell name** field. Edit it to produce a new name for the cell.
- **New cell name:** A new cell name must be entered into this field.
- **Cancel:** Pressing this button abandons the **Cell→Rename** operation.
- **OK:** Pressing this button after selecting a cell name completes the operation.

If you press **OK**, there are two possible outcomes:

1. A message box appears with a warning that such a cell already exists. Click the **OK** button of the message box to repeat cell name selection.
2. The **Cell→Rename...** dialog closes. The operation was performed successfully.

Results

If the operation was successful, the project changes as follows:

- The selected cell is renamed (the cell with the old name does not exist from the moment the operation was completed). All instances of this cell are replaced by instances of the renamed one everywhere in the current project and all activated libraries.

Warning: If a library cell is renamed, its instances will become unresolved in all other libraries that refer to it, but are not activated for the current project at the moment of renaming.

4.12: Importing Cells to Project

Purpose

Adds a cell or a project from file (stored in ELD, SLD, CIF, Applicon, LEF, DEF, GDSII, or OASIS file) to the current project of type eld or sld.

Execution

Click the menu command, **Cell→Import...**. The **File Name Selection Form** appears, titled **Select Project file**.

It has the following fields and buttons:

- **Files of type:** Choose the ELD, SLD, CIF, Applicon, LEF, DEF, GDSII, or OASIS format of the file containing the cells to be loaded.
- **File Name:** The name of the file containing the required cells must be entered into this field.
- **Cancel:** Clicking this button abandons the **Cell→Import...** operation.

- **Open:** Clicking this button after entering a file name starts the actual import operation: the selected file is being read and the list of its cells will appear in the **Import Cell(s)** dialog. This dialog looks the same as **Copy Cell(s)** dialog described in Section 4.9: "†Copying of Cells" and shown in Figure 4-9. Using this dialog, you can import cells from the selected file into the current project or any activated libraries in the same way as cell copying is performed.

In particular, if cell name collision takes place between imported cells and the ones already present in the target project/library, a dialog opens to process name collisions, which is described in Section 4.9.1: "Cell Name Collision Resolving Dialog".

Note: The list of all file names with their extensions existing in the current directory is displayed in the Select Project file window. Double-clicking on a name from this list starts the cell import process immediately, with the current settings for cell import. Single-clicking on a name from this list copies it into the File Name. Edit here to produce a new name.

4.13: †Exporting Cells

Purpose

Saving selected cells into the ELD, SLD, CIF, Applicon, xi-script, OASIS or GDSII file format on disk.

Execution

The execution is perform in two steps:

- first specify which cell(s) need to be exported and the export options,
- then specify the type of export (eld, sld, …) by selecting the file type of the resulting file.

First step:

- Click onto the **Cell→Export...** menu.
- The **Cell Name Selection** box appears, titled **Export Cell(s)**, with the following fields; Library List and Cell List. From the Library List, select the library in which reside the cell(s) to export. From the Cell List select the cell(s) that you want to export. The check boxes **With Hierarchy** and **Multifile Export** are available to customize the type of export desired. These options are described below. When the cell(s) selection is completed, press **OK** or **Apply** to continue the cell(s) export process. Pressing **Cancel** at this stage will abandon the cell export operation.
  - If the **Multifile Export** option is NOT checked, after clicking the **OK** button you are prompted to select a file name to write all selected cells into it.
  - If the **Multifile Export** option is checked, all cells are saved in separate files. In this case, after clicking the **OK** button of the Export Cell(s) dialog, you will be prompted to select a file name prefix, so that each file will have the name: `<prefix><cell_name>.gds`, `<prefix><cell_name>.eld` or with other extensions based on the file type selected.

For each cell exported, the corresponding file(s) may contain the cell(s) and all hierarchy of subcells referenced in it, depending on the status of the **With Hierarchy** check box.

- If the **With Hierarchy** option is NOT checked, the cell(s) exported will contain only the top level, without any subcells. The reference to the subcells will still be present.
- If the **With Hierarchy** option is checked, the cell(s) exported will contain the top level and all hierarchy of subcells referenced in it.
- If the **With Library Cells** is checked, the subcells from linked libraries will be included in the hierarchical cell operation (**With Hierarchy** is on).
- If the **With Library Cells** is not checked, the subcells from linked libraries will be excluded in the hierarchical cell operation (**With Hierarchy** is on).

Second step:

- If **OK** or **Apply** was selected; the **File Name Selection** box appears titled **Select Project file**. Select the type of export file from the drop down menu: GDSII, OASIS, CIF, Applicon, Xi-script, ELD or SLD.

The **File Name** field is used to type in the file name (extension is added automatically).

Alternatively, an existing name may be selected (with the extension) from the list of files. Click on it and it is copied into the **File Name** field. You may edit it there or use as is. Clicking the **Save** button saves the project into the specified file. Clicking the **Cancel** button abandons the save operation.

If the name of the existing file is selected for saving, a warning box appears with three options:

- Pressing the **Yes** button overwrites the existing file.
- Pressing the **No** button returns control to the **Save As** dialog box.
- Pressing the **Cancel** button abandon the save operation.

Results

Saves the cells in the specified file. All cells instanced in the saved one (directly or recursively) are saved into this file as well, unless an option to avoid this is set, as mentioned above. If a cell that contains instances of parameterized cells is exported into a foreign format (GDSII), each instance with a unique set of parameters produces a "personal" cell in the foreign file. Its name is constructed from the name of the P-cell and the values of actual parameters.

4.13.1: Cell Export... to GDSII Files

For example, if you decided to export, in a GDSII format, cells `mux1` and `nand2` from the tutorial example `mux4`. The resultant file would differ in content depending on the options selected.

If **Multifile Export** is NOT selected, the file name chosen is used as is. Let's assume that you chose `mux` as Filename.

- If the **With Hierarchy** and **Multifile Export** options are left unchecked, then EXPERT produces the file `mux.gds`. Only the selected cells are in the GDSII file: `mux1` and `nand2`. In this case, as `nand2` is an instance in `mux1`, the `mux1` cell will display `nand2` but not its inverter cell.
- If only **With Hierarchy** is checked, then EXPERT produces the file `mux.gds`. The GDSII file contains the cells selected (`mux1`, `nand2`) and the cells of the instances used at the top level (`inverter`).

If **Multifile Export** is selected the file name chosen is used as the prefix. Let's assume that you chose `"new~"` as a prefix.

- If **With Hierarchy** and **Multifile Export** options are checked, then EXPERT produces the files, `new~mux1.gds` and `new~nand2.gds`. Both files contain the top level cells, and the cell(s) of the instances used by the top level.
- If only **Multifile Export** is checked, then EXPERT produces the files, `new~mux1.gds` and `new~nand2.gds`. Both files contain one cell with only the top level geometries.

Note: If the **With Hierarchy** option is left unchecked, then only the selected cells will be written in the GDSII file. If these cells contain references to some other cells, they will be unresolved in the resulting GDSII file.This option is useful for splitting a project into several libraries.

4.13.2: Cell Export... to Xi-Script Files

The menu command, **Cell→Export...** can be used to export in **Xi-Script** file format. When this operation is performed, EXPERT writes two files:

- A Xi-Script file with the definition of the cell selected for export;

- A technology file saved from the current project. You may see the name of this tcn file used in the first line of the exported xi-script as a parameter to the command `project new`.

4.14: Saving /Discarding/Forbidding Cell Changes

The following commands from cell submenu handle changes in cells:

Cell→Save, **Cell→Save Cell As...**, **Cell→Discard Changes**, **Cell→Read Only**.

See also:

- **File→Save**, **File→Save As...**, **File→Save Cells...**.
- **Edit→Undo**, **Edit→Redo**.
- Commands from Chapter 16: "Libraries and Version Control for a Multi-User Design Environment".

4.14.1: Save

The **Cell→Save** command saves all changes in the current cell into the corresponding project/library file.

4.14.2: Save As...

The **Cell→Save Cell As...** command writes the current cell together with all edits into a new cell with another name. All recent changes (i.e., the ones performed after the last save command) in the current cell are discarded.

4.14.3: Discard Changes

The **Cell→Discard Changes** command discards all recent changes (i.e., the ones performed after the last **File→Save**, **File→Save Cells...** or **Cell→Save** command) in the current cell are discarded.

4.14.4: Read Only

If the cell has no unsaved changes, then the **Cell→Read Only** command sets/removes the read-only status for the current cell. This status means that the cell cannot be edited.

Notice that cell may have read-only status for other reasons:

- The project/library may be of read-only type.
- The project/library may be of the Shared Layout data format type and the cell is not checked out.
- The project/library may be opened in read-only mode. In the current versions of EXPERT, it happens if another copy of EXPERT has already opened the project in read/write mode.

4.15: Cell Information

Selecting **Cell→Info...** and **Cell→Selection Info...** menu commands report various data about the cell, including:

- EXPERT version used;
- current time;
- current user;
- current project;
- cell name;
- its type (top cell, leaf cell, read-only cell, etc.);
- date of creation;
- date of modification;
- size of the current cell;

- its shape domain (coordinates of the lower left and upper right corners of the minimal bounding box) and drawn domain (total domain of shapes and texts);
- total object count in the cell and the list of them;
- child cells hierarchy statistics.

This last hierarchy statistics is especially useful when you are at the top cell level as it will give you a view and a count of all cells used in your project. Figure 4-11 illustrates the **Cell Info** window for a given cell.

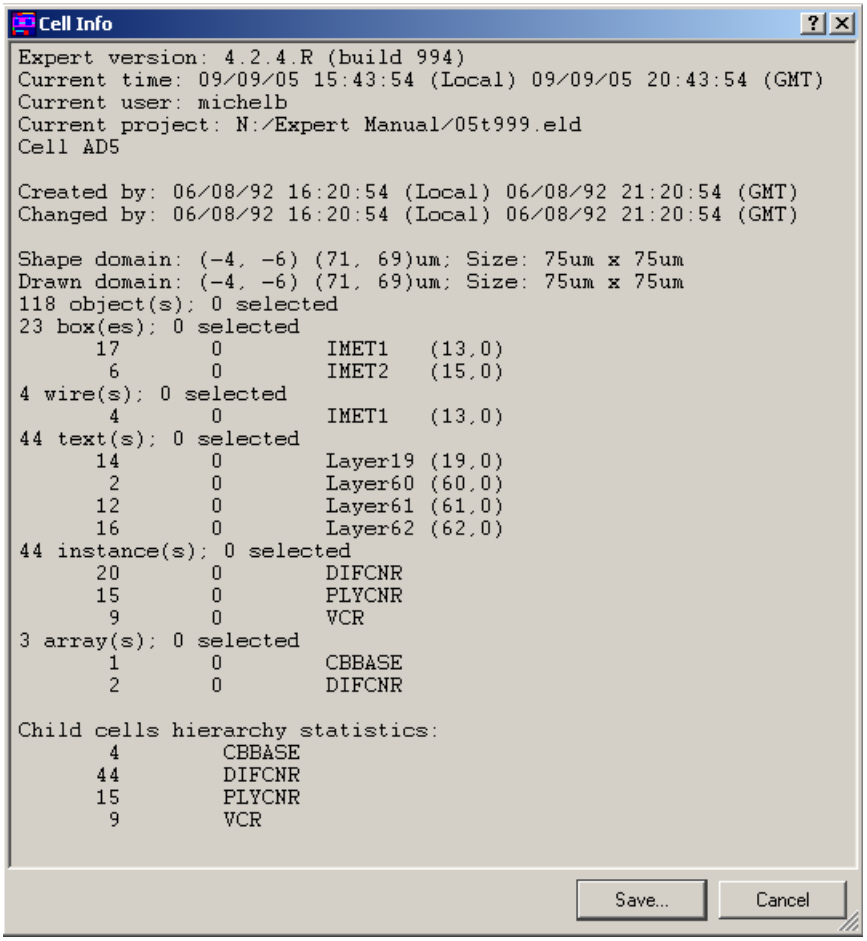

Cell Info
Expert version: 4.2.4.R (build 994)
Current time: 09/09/05 15:43:54 (Local) 09/09/05 20:43:54 (GMT)
Current user: michelb
Current project: N:/Expert Manual/05t999.eld
Cell AD5
Created by: 06/08/92 16:20:54 (Local) 06/08/92 21:20:54 (GMT)
Changed by: 06/08/92 16:20:54 (Local) 06/08/92 21:20:54 (GMT)
Shape domain: (-4, -6) (71, 69)um; Size: 75um x 75um
Drawn domain: (-4, -6) (71, 69)um; Size: 75um x 75um
118 object(s); 0 selected
23 box(es); 0 selected
17 0 IMET1 (13,0)
6 0 IMET2 (15,0)
4 wire(s); 0 selected
4 0 IMET1 (13,0)
44 text(s); 0 selected
14 0 Layer19 (19,0)
2 0 Layer60 (60,0)
12 0 Layer61 (61,0)
16 0 Layer62 (62,0)
44 instance(s); 0 selected
20 0 DIFCNR
15 0 PLYCNR
9 0 VCR
3 array(s); 0 selected
1 0 CBBASE
2 0 DIFCNR
Child cells hierarchy statistics:
4 CBBASE
44 DIFCNR
15 PLYCNR
9 VCR
Save...
Cancel

Figure 4-11: Cell Info

When a cell is loaded from the GDSII file and it contains an invalid date, it is set to the current date.

The information about objects is grouped by object types and layers. It is displayed in a text window from which it may be cut and pasted into any Windows application (e.g., into a word processor). To do this, select the required part of the following information using the mouse.

- Depress the left button with the cursor at the beginning of the required piece.
- Drag cursor across the text and release the button at the end of the required piece.
- Press **Ctrl-C** to copy the text into Windows NT buffer.
- Switch to your word processor and press **Ctrl-V**.

You can also save all information into a file by clicking the **Save** button.

Hint: If you want a detailed list of cell objects, together with all their coordinates, and so on, select **Cell→Export→...,Xi-Script File**.

Hint: If you want detailed information for selected objects, you can create a temporary cell from the selected objects, export it into xi-script, and then delete it. In fact, you can write a short xi-script to do this for you.

This page is intentionally left blank.

Chapter 5:
Creating Layout Objects

5.1: Creating Layouts

This chapter describes how to create layout objects: wires, regions, cell instances/arrays; and annotation elements: texts, or labels. Wires can be of several styles. Some special region types: boxes, ellipses/circles, donuts (rings) and rolled regions can be created by separate means. In fact, a wire is also a special kind of region.

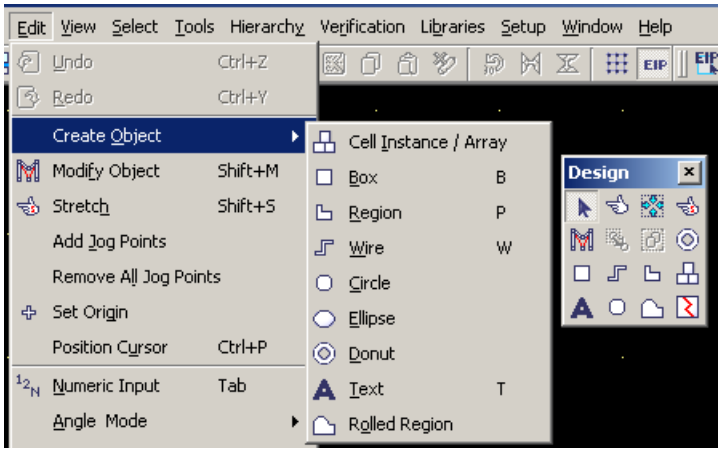


Figure 5-1: Object Creation Menu and Tool Bar

EXPERT gives the following possibilities:

- Starting object drawing from the menu in the **Design Bar** through a keyboard shortcut (bindkey) or from the xi-command line.
- Creating primitive objects (i.e., other than instances/arrays) in the **active layer**. To make a layer **active**, either click at the color strip of the layer in the **Layer Bar**, or select it in the **Numeric Input** panel.
- Placing restrictions on the geometry is created:
  - by snapping the cursor to grid,
  - by selecting admissible angles,
  - by selecting a direction snap mode.
- The newly-created shapes will modify the existing ones if you set the **hole** or **merge** mode of a primitive object creation.
- Creating objects by using point-and-click mouse actions, precision numeric input of geometric parameters, or a combination of both.
- Creating objects by running DRC scripts, xi-scripts, or by commands from the xi-command line.

Note: You can use the **Property Bar** instead of the **Numeric Input** panel for inspecting/modifying attributes of objects while they are created. In order to do so you need to select option at **Setup→Editor/Viewer Setup→Editing→Use Property bar for Numeric Input**. You can read more about the Property Bar in the Section 8.9: "Inspecting/Changing Object Properties".

5.2: Drawing Settings

5.2.1: Admissible Angles

The **Angle Modes** submenu and the corresponding buttons in the **Drawing Bar** are used to specify admissible directions of region sides during their creation.

Under the 90° restriction, region sides and wire segments can be only vertical or horizontal. Two more diagonal directions are allowed under the 45° restriction (Figure 5-2).

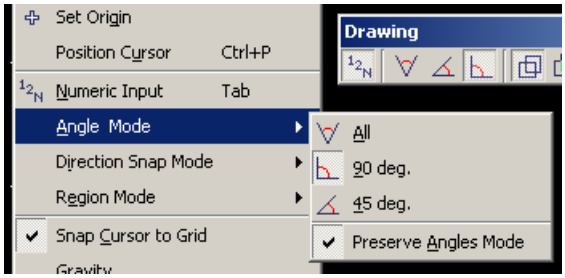


Figure 5-2: Admissible Angles: Menu and Toolbar

5.2.2: Snap Cursor to Grid

In this mode, the cursor can be positioned only at the nodes of the snap grid. It can differ from the visual grid. To set grid parameters, select **Setup→Technology→Grid**. See Chapter 11: "Setup" for the description of grids used in EXPERT.

5.2.3: Direction Snap Modes

These modes restrict the cursor's movement to specific sets of directions during the creation and modification of objects. To use these modes, either select **Edit→Direction Snap Modes**, or click the buttons at the **Drawing Bar**.

The following modes are available:

- None (no restrictions on cursor movements)
- Diagonal (cursor may move in vertical, horizontal, and diagonal directions from the starting point)
- Orthogonal
- Vertical
- Horizontal

5.2.4: Region Modes

When you create a new shape, you can use it to modify existing ones by selecting one of the three **Region Modes** from the menu or from the **Drawing Bar.** These modes are **Normal**, **Hole**, or **Merge**. These modes are mutually exclusive.

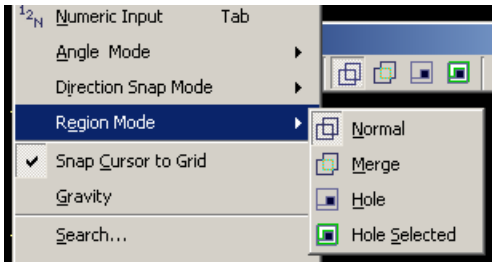


Figure 5-3: Region Mode Submenu

Normal mode simply draws a new shape, Hole mode is for making holes, and Merge mode is for merging shapes.

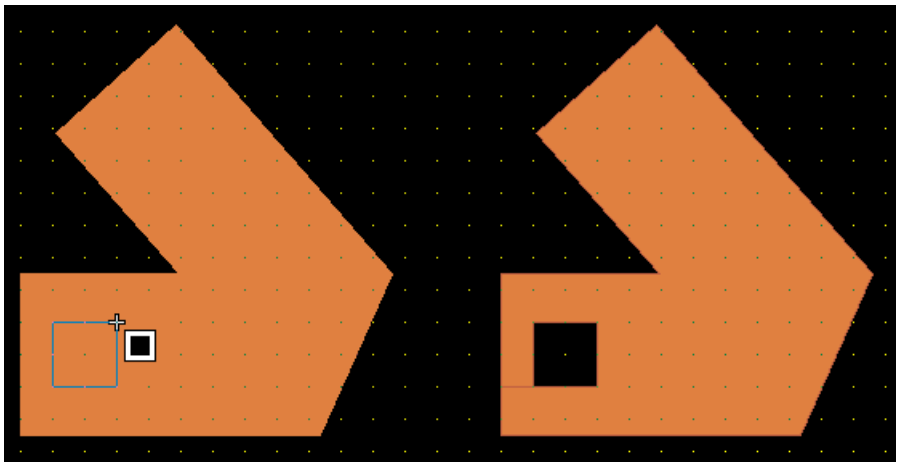


Figure 5-4: Creation of a Box Hole in Wire

Hole Mode

In this mode, the new object is treated as a hole or as a cut in the existing ones in the active layer. If the new object overlaps several old ones in the active layer, it makes holes in all of them.

- If the new object is completely inside an old one, then a true hole is created. In Figure 5-5, a new **box** creates a hole in a thick **wire**. Note that in this case the **wire** is converted to **region**.
- If the new object partially overlaps an older one, then a piece is cut off from the old one. In Figure 5-5, a new **wire** cuts into old boxes (and even splits one of them).
- If the new object is completely outside of all older ones, then nothing is created.
- If the new object completely overlays an older object, then the old one is deleted.

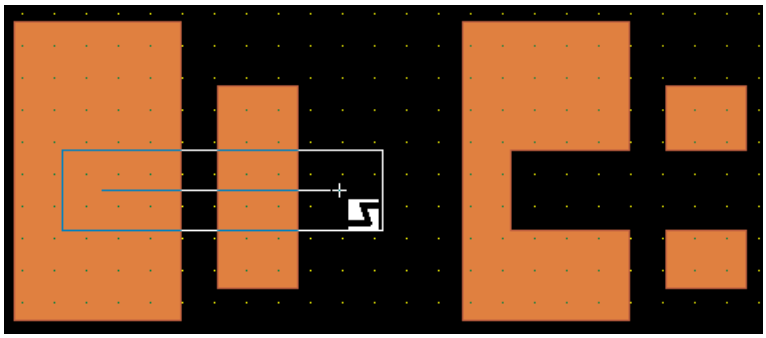


Figure 5-5: Wire Drawn in Hole Mode

Hole Selected Mode

In this mode, the new object is treated as a hole or as a cut in the selected ones, possibly in different layers. Other than that, this mode is similar to Hole mode.

Merge Mode

If a new object is created in this mode and it intersects older ones, then these objects are converted into regions and a **merge** (union) operation is performed with them (see Figure 5-6).

.

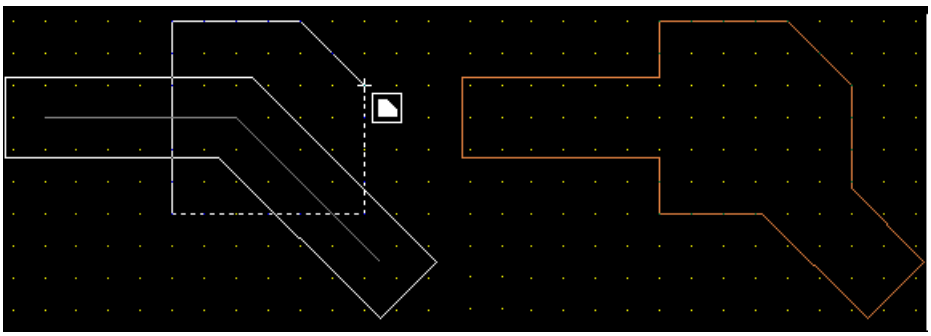


Figure 5-6: New 45-mode Region is Merged with Old Wire

Over the Point Drawing

Over the Point Drawing is used to reduce the amount of clicks needed to create a polygon or wire. In other words, you only need to click the mouse button once to create two consecutive segments of an object. The Over the Point Drawing mode works if either of two conditions are met:

- the Angle Mode is set to 90 deg. and Direction Snap Mode is set to Diagonal or None
- Direction Snap Mode is set to Orthogonal

To use this mode, set the **Allow L-Shape Mode input for regions and wires** option in the Editor/Viewer Setup Dialog Box (**Setup→Editor/Viewer→Editing**).

If the **Vertical First** option is used (**Edit→Region Mode→Vertical First**), the first segment of the L-shape is vertical and the second one is horizontal. If this option isn't used, the first segment of the L-shape is horizontal and the second one is vertical. See Figure 5-7.

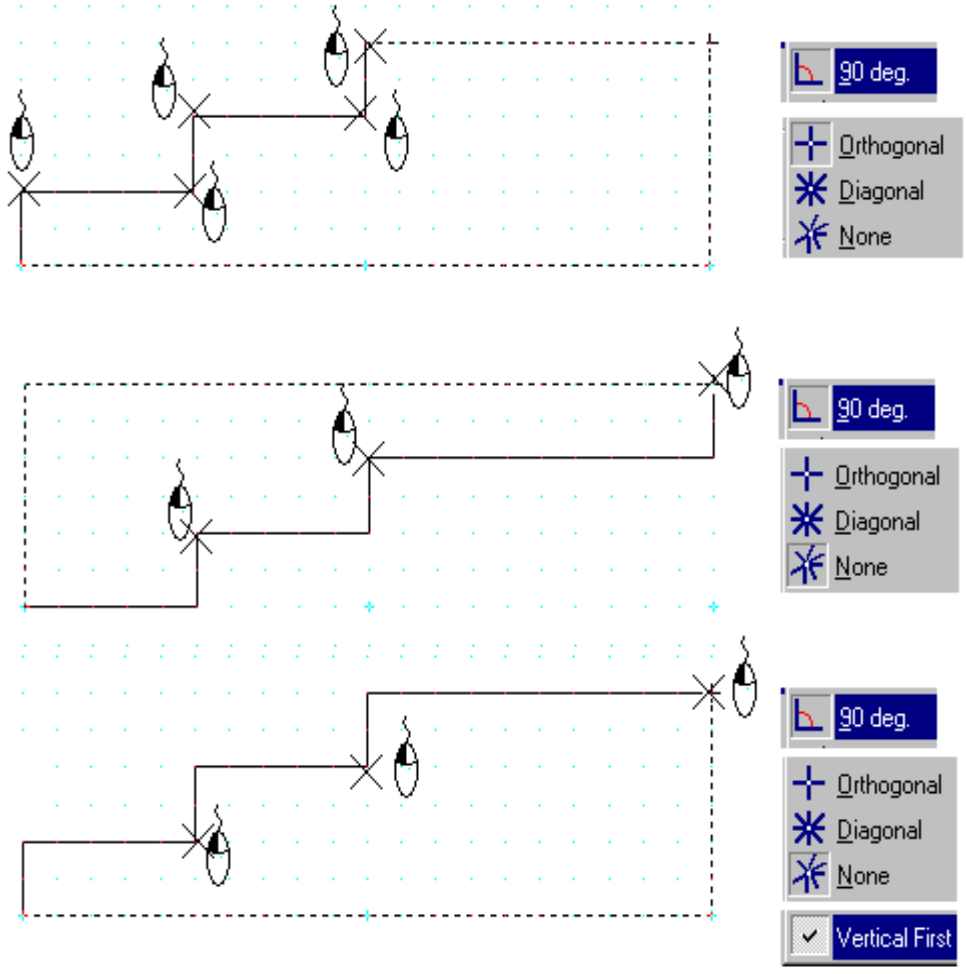


Figure 5-7: Polygonal Creation in Over the Point Drawing

5.3: Cell Instance/Array Creation

Cell instances and arrays are created by the same tool. To create them, select **Edit→Create Object→Cell Instance/Array** or use the corresponding buttons from the **Design Bar**. The **Numeric Input** panel will appear. The cursor shape will change to the instancing cursor with the instance frame attached. This frame has the dimensions of the domain of the instantiated cell. The cursor position, with respect to the instance frame, indicates the reference point or origin of the activated cell.

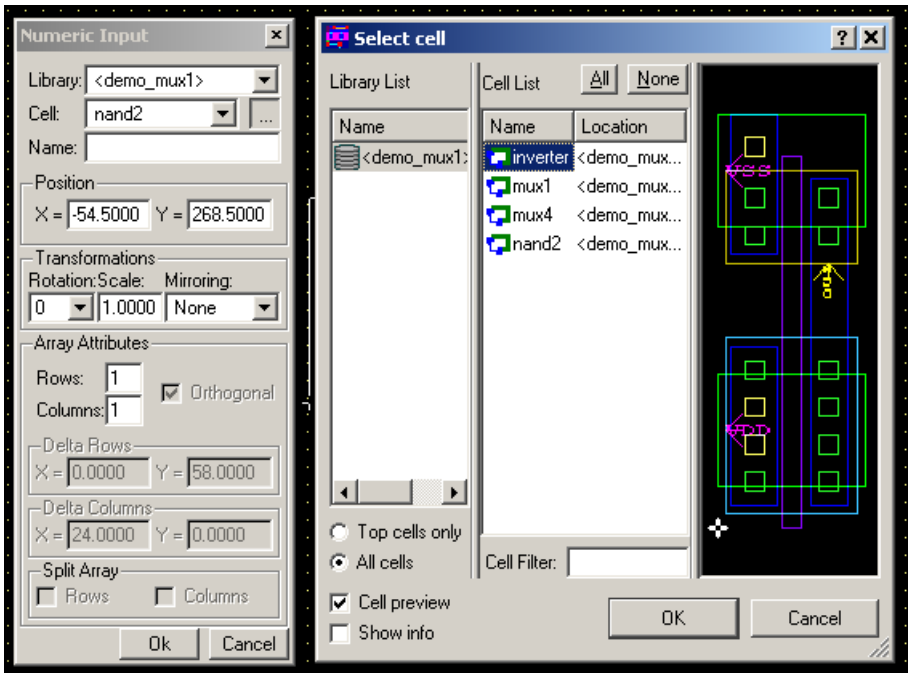


Figure 5-8: Cell Instancing Dialog, together with Cell Selector Dialog

You can enter/edit instance/array attributes in any order. Two properties are mandatory: `cell selection` and `instance position`. The remaining ones are optional.

To finish cell instance creation, click the **OK** button in the **Numeric Input**, or click the left mouse button anywhere in the layout.

5.3.1: Instance/Array Attributes

Cell: The drop-down lists **Library Name** and **Cell Name** are used to select the cell to be placed. Alternatively, you can click the browser button (marked with **...**) and use the standard cell selection dialog. **Cell** selector stops the selection of cells that may produce recursive instancing.

Position: The coordinates of the position of the instance/array can be assigned both by mouse and by numeric input.

Hint: When working under low resolution, clicking the mouse creates instance in an approximate position. This instance can then be selected and its position can be fine-tuned by numeric input in the Property Bar.

Note: For instances of small visible size, cell origins are not shown even if you check the corresponding option **Setup→Editor/Viewer...→Viewing→Show cell instance origin**.

Rotation: The corresponding drop-down list contains four orthogonal rotation angles: 0°, 90°, 180°, and 270°. In addition, the angle can be set manually to any value.

Note: If the number of rows or columns (or both) is not equal to one, then only orthogonal rotation angles are allowed. In other words, you cannot rotate an array by an arbitrary angle.

Scaling: Any positive real number is allowed.

Mirroring: Three choices are available from the corresponding drop-down list: **None**, **Vertical**, and **Horizontal**. Mirroring is performed with respect to the axis passing through the position point of the cell instance. Vertical mirroring is done in a vertical direction. In other words, verticle mirroring is done with respect to X-axis, and horizontal mirroring is done with respect to Y-axis.

Note: When the instance creation/modification is finished, horizontal mirroring is replaced by the equivalent combination of vertical mirroring and rotation by 180°.

Clicking the **OK** button in the **Numeric Input**, or clicking the left mouse button in the Layout area completes the creation of one instance/array, and immediately starts the creation of the next cell instance.

Array creation: An array is created if the number of rows or columns (or both) is greater than the one entered. The frame attached to the cursor turns into the corresponding array of frames, allowing you to enter relative displacements of array entries (row and column **deltas**). There are two modes for deltas: **orthogonal** and **arbitrary**. The difference is that the orthogonal displaces rows only in the Y direction, and arbitrary displaces columns only in the X direction.

Note: If a non-orthogonal rotation angle is given, an array cannot be created.

In addition, there is the **Split Array** control group with two check boxes: **Split Rows** and **Split Columns**. If both boxes are checked, the array will be split into separate instances. If only the **Split Rows** box is checked, the array will be split into row-arrays. If only the **Split Columns** box is checked, the array will be split into column-arrays.

5.3.2: Instance of Parameterized Cell

If you select a parameterized cell for instancing, then the **Numeric Input** contains the list of parameters for corresponding cells. Parameters are initialized with default values. You can change value of any parameter in the list.

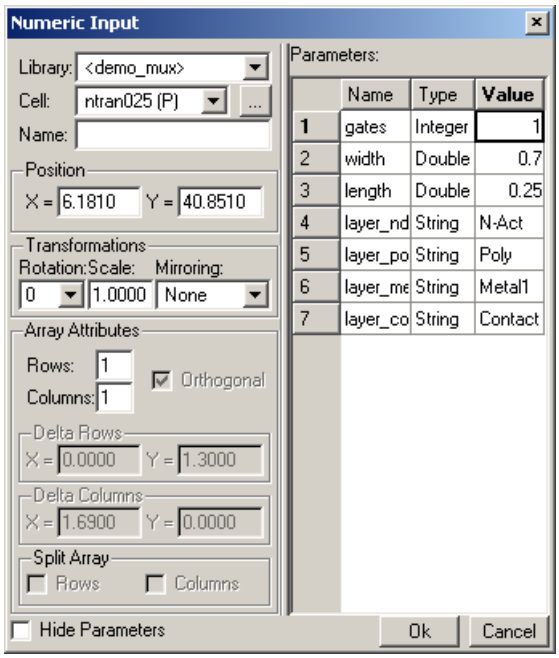


Figure 5-9: Parameterized Cell Instancing

5.4: Primitive Object Creation

Primitive objects (shapes and texts) are created in the **active layer**. Therefore, before object creation, the necessary layer must be set to **active**. To do this, perform one of the following steps:

- Click in the color strip near the required layer name at the **Layer Bar** (recall that clicking the layer name shows/hides this layer),
- Select **Tools→Pick Layer**. See Chapter 12: “Tools”, Section 12.16.1: “Scratch Layers” for more information.
- Select active layer for the corresponding tool in the **Numeric Input**.

If the required layer is absent in the layer list, you must create it by selecting **Setup→Technology→Layer/Wires Setup...** .

The shapes of created objects and the way they are created depend upon the following settings: **Region Mode**, **Angle Mode**, **Numeric Input Mode**, and **Cursor snapping**, described above.

By default, object creation operations are **modal** ones. for example, when you select the box creation command box, enter the **box creation mode** and you can create several boxes until you switch to some other operation or to the **void tool**. To start an object creation mode, either select **Edit→Create Object** or click the corresponding button from the **Design Bar**. This mode is recognizable by its cursor shape, which is similar to that drawn on the **Design Bar** button.

5.4.1: Polygonal Region Creation

Default Way of Execution

Point the cursor at the required region vertices one by one and click the mouse on them. All vertices, except the last one, must be clicked by the **left** mouse button. The **last** vertex must be clicked by the **right** mouse button.

A click of the right mouse button means that the region is complete. You can then start creating a new one, or change the tool to do something else.

After entering the first two segments, the region is always closed during creation. The last closing side (or sides, if you are in 90° or 45° angle mode) will be drawn dashed. It becomes solid when creation is completed.

Note: If you create a polygon or a wire, the info in the message line shows the region’s area coordinates, displacements of the cursor with respect to the previous point, and the number of vertices as follows: “...N=46...”.

Variants

Sometimes, it is inconvenient to use **right** mouse click to enter the last vertex of the region. There are two ways to do without this.

1. Select **Tools→Mouse Click→Right** to assign a bindkey to replace the right mouse click by hitting a key.
2. Select **Setup→Editor/Viewer...→Editing** and the Last mouse click does **not** add a point to wire/region.

Numeric Input

You can use **Numeric Input** to perform precision input of vertices at any place of the vertex list, and even modify already created vertices. Also, you can combine mouse input and numeric input.

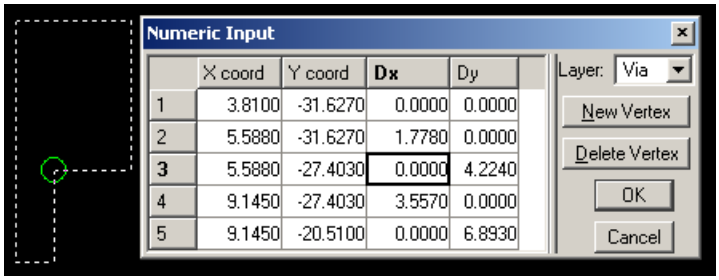


Figure 5-10: Region Numeric Input

- **New vertex:** Clicking this button adds a new vertex to the list.

The new vertex can be specified either by its **X** or **Y** coordinates, or by displacements **Dx** and **Dy** with respect to the previous vertex.

- **Delete vertex:** Clicking this button deletes the highlighted vertex.

5.4.2: Box Region Creation

This is the simplest operation, done either by two mouse clicks or by mouse drag.

1. Click the mouse at the two required opposite corners of the box.
2. Alternatively, press the left mouse button at one corner, and drag to the other one and release the button.

Numeric Input Mode

You can use the **Numeric Input** to precisely define position and size of a new box. Clicking the **OK** button in the **Numeric Input** finishes box creation.

Notice that the message line shows box area and dimensions.

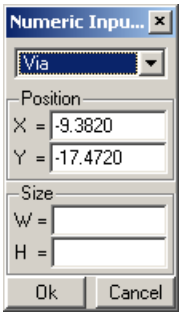


Figure 5-11: Box Numeric Input

Hint: If you want to create a square box with the mouse, use the diagonal direction snap mode (**Edit→Direction Snap Mode→Diagonal**).

5.4.3: Circle/Ellipse Creation

The first mouse click defines the center. The second mouse click defines both radii of the ellipse, or the radius of the circle.

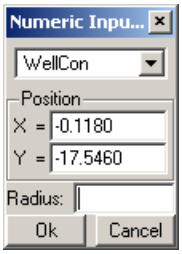


Figure 5-12: Circle/Ellipse Numeric Input

Numeric Input Mode

Properties for circle/ellipse are the same as for box.

The circle/ellipse is converted into a polygon according to the current settings from **Setup→Technology→General...→Shapes** in the following cases:

- When selecting **Setup→Editor/Viewer...→Editing→Convert ellipses/circles to polygons immediately**
- When created in hole or merge mode
- When selecting **Tools→Merge Selected**
- During export into GDSII format
- During internal conversion for DRC operations

Otherwise, in the normal mode of object creation, a circle/ellipse is drawn and stored in the database as a special object.

Hint: While in the "Ellipse" mode, if you want to create a **circle** rather than an arbitrary ellipse by mouse, use the diagonal direction snap mode (**Edit→Direction Snap Mode→Diagonal**) during ellipse creation.

5.4.4: Donut (Circular Ring) Shape Creation

A donut (circular ring) is a polygon approximating the region between two concentric circles. The precision of approximation is specified in **Setup→Editor/Viewer...→Shapes**.

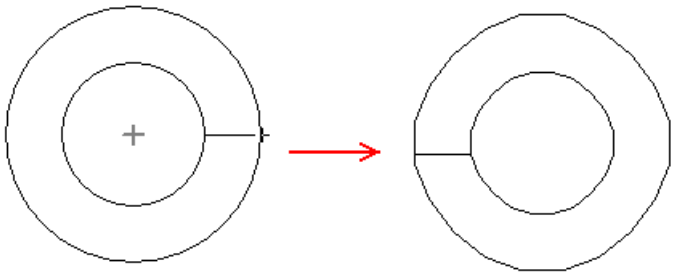


Figure 5-13: Donut

Execution

The first mouse click defines the center. The second and the third mouse clicks define both radii of the circular donut.

As in other shapes, donuts can also be created in hole and merge modes.

During creation, the message line shows donut radii and area.

During creation, the donut is drawn as exact concentric circles, as shown on the left side of Figure 5-13. After creation, the donut is converted into a polygon as shown on the right side of Figure 5-13, according to the settings in **Setup→Technology→General...→Shapes**.

5.4.5: Rolled Region Creation

Purpose

Rolled region is created in the same way as a polygonal region, except the radius of curving is defined for each vertex. The radius can be changed during drawing. Changes of the radius affect only joints of the last segment. All points input before the last one preserves their settings.

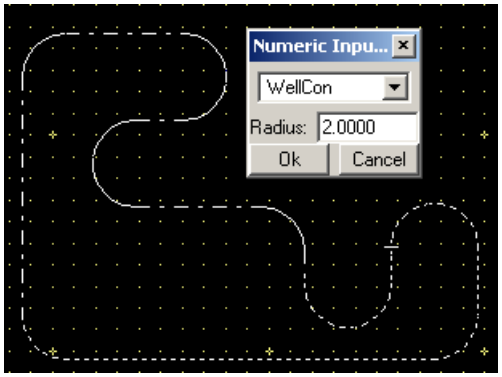


Figure 5-14: Rolled Region

5.4.6: Wire Creation

To create a **wire**, first define the following parameters:

- **Active layer:** The layer where all new objects are created. To create a wire in a layer, set to **active** as described in Section 5.4: “Primitive Object Creation”.
- Every layer, when defined in the technology, has default values for the following wire parameters:
  - wire width
  - end style
  - joint style

Select **Setup→Technology→Layer/Wire Setup...** to set (or change) them. You can override the defaults for the new wire in the **Numeric Input**.

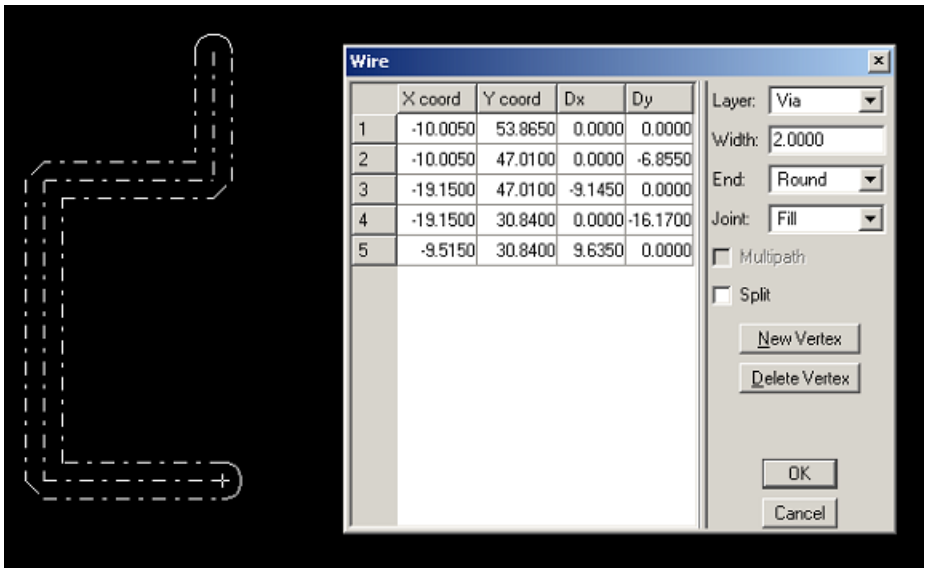


Figure 5-15: Wire Creation

When you are drawing geometry on a grid, there is a natural restriction on the wire width. It must be an even multiple of the minimum grid step. If you are trying to set a wire width that does not satisfy this condition, a warning message appears.

During wire creation, the message line shows wire length and displacements of the cursor with respect to the previous point.

The construction of a wire is performed similarly to that of a region, including numeric input of vertices in the **Numeric Input**. The exception is that the mouse clicks define the wire’s centerline rather than the region’s boundary.

Zero-Width Wires

You may need to make some marks in the layout (e.g., mark centerlines or alignment) with lines. In other words, using one-dimensional objects. It is recomended that you use wires of zero width. Boxes and polygons of zero thickness are not allowed, even if you import them from a GDSII file. They will be purged away either during editing operations or during saving.

Off-Grid Wire Shapes

To use this option, select **Setup→Technology→General...→Shapes→Offgrid wire boundary allowed**.

- If this setting is **off**, a 45° mode wire is corrected in such a way that the vertices of its boundary polygon fit on the minimal grid. The width of diagonal wire segments will then increase.
- If this setting is **on**, the wire shape will be drawn according to its centerline and (constant) width.

This option affects the drawing of wires, and the **Convert to polygon** operation for wires.

Warning: If this option is **on**, you must use the offgrid checks in DRC for the final design.

Multiwire

Multiwire is a complex layout object that consists of several wires running in different layers consecutively connected by appropriate contacts. Creation of multiwires is disabled if technology doesn't have defined contacts. See Section 11.4.8: "Contact Setup" to see how contacts are defined.

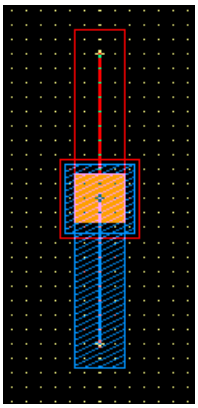


Figure 5-16: Multiwire

Multiwires are created in a similar way to wires. Simply check the **Multipath** checkbox in the Numeric input panel, and switch the layers when you want the contact to be placed.

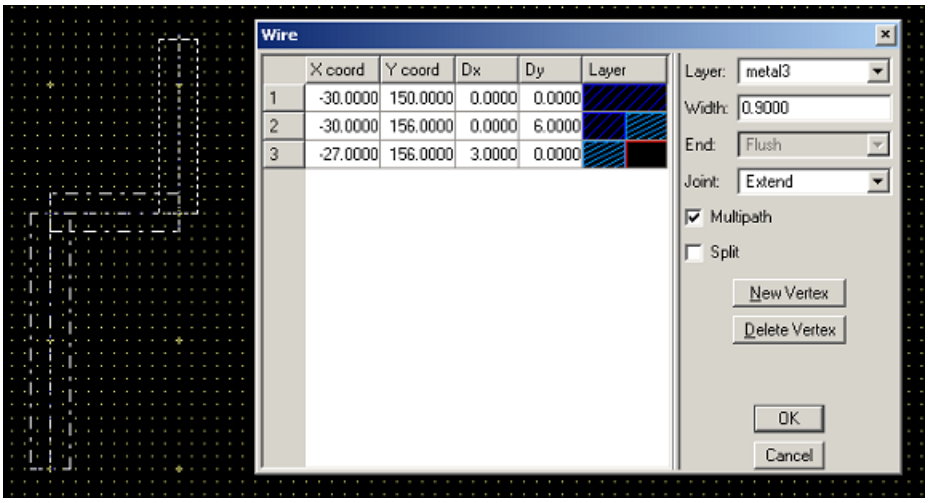


Figure 5-17: Multiwire Creation

A sample drawing of a Multiwire shown in Figure 5-17 may look like this:

- Check the **Multipath** checkbox.
- Select **Layer metal1** and click two points to define the first wire segment.
- In the Numeric input panel, change **Layer** to **metal2** and click one point to define second multiwire segment.
- Change **Layer** to **metal3** and right-click one point to finish the multiwire creation.

Multiwire segments on different layers can have different width and styles. You can change these settings after you switched to appropriate layer, or you can use left part of the Numeric input panel to highlight vertices corresponding to segments you want to change.

By default, all parts of Multiwire are united together into one group. Thus multiwire could be edited in the same way as a regular wire. However, if you want them to be created as separate objects, check the **Split** checkbox in the Numeric input panel of the **Wire** tool.

5.4.7: Rolled Wire Creation

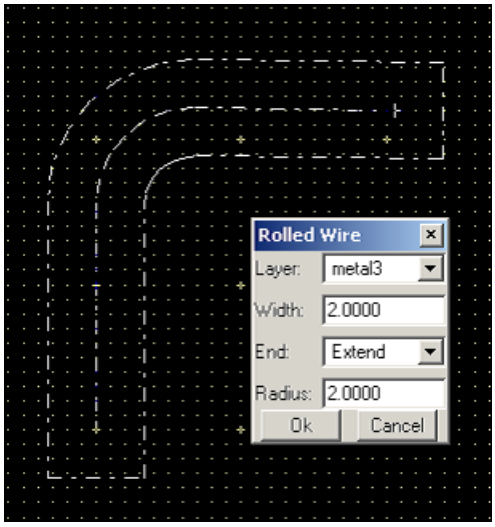


Figure 5-18: Rolled Wire Creation

Rolled wire is created in a similar way to a regular wire, except in addition to the usual parameters, you need to define the radius of curving.

Layer, **Wire Width** and **Wire End Style** parameters are the same as in a regular wire. **Wire Joint Style** is predefined (set to **Extend**) and cannot be changed.

5.5: Text/Label Creation

Selecting **Edit→Create Object→Text** allows you to write one-line **text** anywhere in the layout. However, only one font is available of any size and text line orientation. In addition, the reference point (origin) of a Text object is shown as “bird's footmark” of constant pixel size.

When a cell instance that contains text is transformed, or the text itself is transformed by the edit commands **Rotate** or **Mirror**, the drawing of a text is transformed as follows:

- its origin and bounding box are transformed into correct positions.
- the text is written inside its bounding box to be the most readable.

In other words, EXPERT does not mirror it nor does it write upside down.

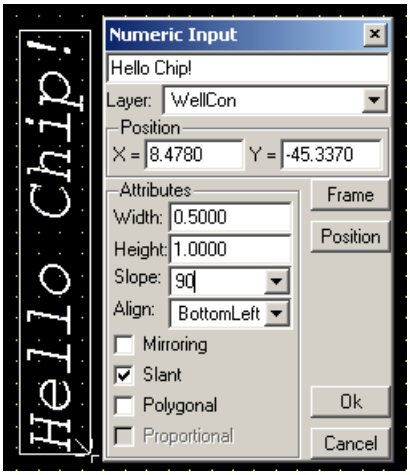


Figure 5-19: Text/Label Creation

To create the required text type in the **Numeric Input**, click the mouse at the desired position. The text will be created with a default size.

To create text with different sizes, enter corresponding values in the **Numeric Input**. Alternatively, you can define the text's frame by mouse. To do this, press the **Frame** button in the **Numeric Input**. When the **Make Frame** cursor appears, drag the frame to a desired size in the layout.

The **Numeric Input** for **Text** creation provides the following properties:

- **Text input field** inputs an "as is" single-line text. In other words, no special character translation is performed.
- Text can be aligned with respect to its position in nine ways by selecting the required alignment from the corresponding drop-down list titled **Align**.
- Text can be **slanted** by checking the corresponding box.
- The **slope** of the text line is set in degrees.

Note: It is impossible to select a font number for texts. However, if a text object is read from a GDSII file, its font number is preserved and may be written back into a GDSII file. Otherwise, the font number is set to 0.

- The **position** of the anchor point.
- The **size** of average text character.

Normally, text is created in the current **layer**, but you may change the layer for the text.

Click **OK** in the **Numeric Input** in order to create text in the current position with current settings.

5.6: Parametric Wire

You can create a Pwire object using the EXPERT menu command **Edit→Create Object→Parametric Wire** or the Xi command `PWIRE`. Once you selected the command, the **Numeric Input** panel pops up. This form is similar to the ordinary wire's Numeric input, but has a couple of additional fields and buttons.

- **Template** allows you to select a template stored in the technology with predefined pwire parameters
- **Name** is used to assign a unique name for the created pwire object. If **Name** is not specified, then a system-defined unique name will be assigned.

- **Subparts** button opens the **Subparts** dialog panel, which allows you to add, remove or update the subparts of a pwire object (see Figure 5-20)

The input fields **Layer**, **Width**, **End** and **Joint** define attributes of the master wire. Vertices for the master wire can be added by mouse clicking or using the **New Vertex** button.

If you are using an existing template from the **Template** drop down list, then parameter values for the path and all subparts will be filled automatically. You can make any changes to predefined parameters, or create a new Pwire from scratch without using templates.

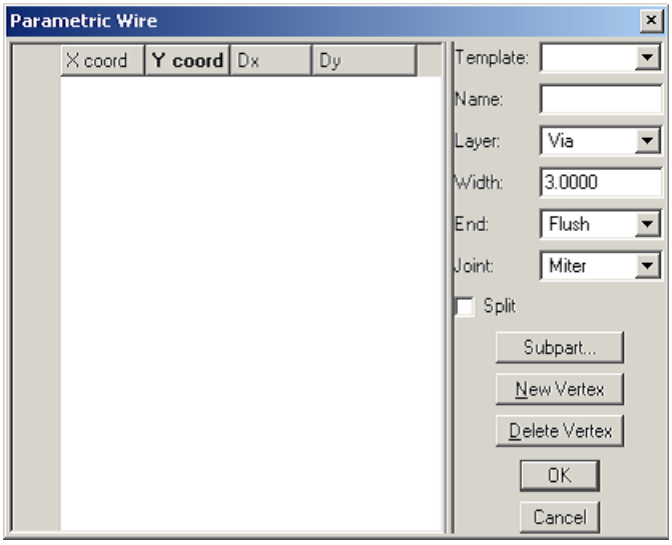


Figure 5-20: Numeric Input

The **Subparts** dialog allows the adding, removing and modifying the subparts for the current Pwire.

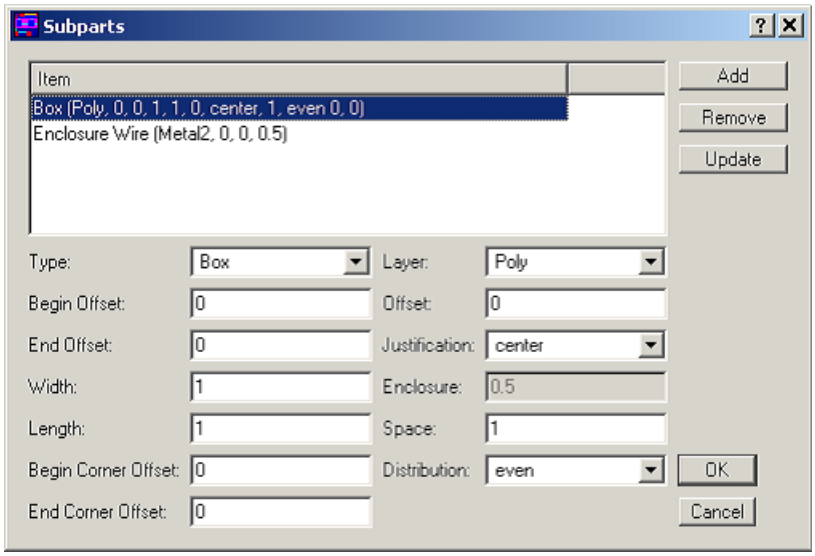


Figure 5-21: Subparts Dialog

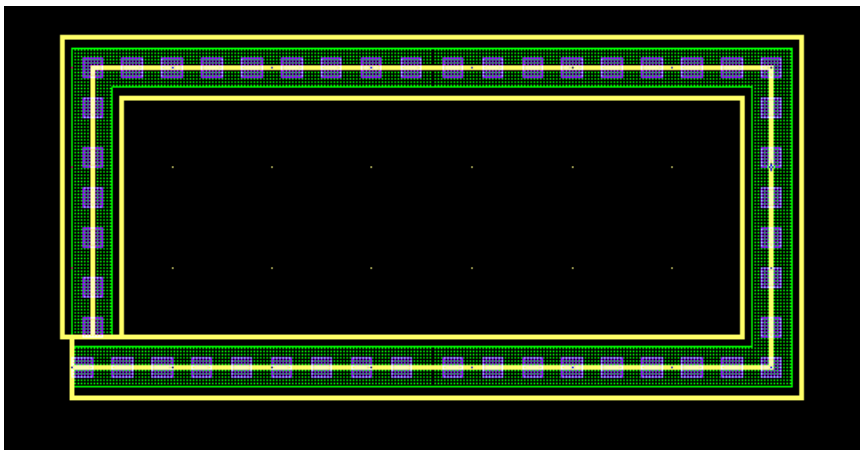


Figure 5-22: Example of PWire

A Pwire object can include any number of subparts, such as enclosure wires, offset wires and sets of rectangles. A Pwire subparts list is shown at the top of the **Subparts** dialog (see Figure 5-21). When you select an existing subpart, the values of its parameters will be automatically filled in the corresponding fields. You can edit the values of parameters, and click the **Update** button to apply the changes to the selected subpart. The **Delete** button is to delete highlighted subpart.

To add a new subpart, first of all, select subpart type: Enclosure Path, Offset Path, or Subrectangle. Then set up specifics for the selected subpart type parameters and click **Add**. A new subpart will be added to the current Pwire, and the subpart list will be updated. Click **OK** to apply changes to the subparts, or **Cancel** to discard any changes.

You can create following types of Pwire subparts:

- **Enclosure Wire** is a wire with vertices coincided with vertices of the master wire. Width is calculated by the formula:

  ```
  width = master_wire_width - (2 * enclosure)
  ```

  where enclosure is defined in the **Enclosure** field of the **Subparts** dialog. You can choose the layer (from the drop down list **Layer**) for the Enclosure path. In addition, you can specify how the starting and ending points of the Enclosure path are located, in respect to starting and ending points of the master wire. Use the **Begin Offset** and **End Offset** fields from **Subparts** dialog to change the beginning and ending points of the Enclosure path. Starting and ending edges of the enclosure path are extended by specified values if the offset values are positive, and truncated if negative.
- **Offset Wire** is a wire with edges parallel to the master wire edges, and shift defined by the **Offset** and **Justification** parameters (see Table 1). The width of the offset wire is defined by the **Width** parameter. Use the drop down list **Layer** from the **Subparts** dialog to set the layer for the offset wire. In addition, you can specify the **Begin Offset** and **End Offset** parameters to extend or truncate the beginning and ending wire segments correspondingly (see description above).

Table 1: Offset Path Separation

Offset	Center	Left	Right
0	Vertices coincide	Left edge master wire coincide with right edge of offset wire	Right edge master wire coincide with left edge of offset wire
>0	Shift left (separation value is distance between center-lines)	Shift left (separation value is distance between left edge of master wire and right edge of offset wire)	Shift right (separation value is distance between right edge of master wire and left edge of offset wire)
<0	Shift right (separation value is distance between center-lines)	Shift right (separation value is distance between left edge of master wire and right edge of offset wire)	Shift left (separation value is distance between right edge of master wire and left edge of offset wire)

- **Box** is the set of boxes laying in the specified layer. The size of the box is defined by the **Width** and **Length** parameters, where box width is measured across the master wire, and box length along the master wire centerline. The distance between boxes is defined by the **Space** and **Distribution** parameters. Space defines minimal distance between boxes. Distribution can be one of two types: **minimum** or **even**. **minimum** means that distance between the boxes is fixed and equal to the **Space** parameter value. Remaining space after the last placed box (if there is one) is left empty. If you chose **even** distribution, then unused space is evenly distributed between placed boxes, and actual space between boxes is in the range:

  ```
  space value < actual space < space value + box length.
  ```

The **Offset** and **Justification** parameters are used to place boxes with offset from the master wire (see Table 2).

Table 2: Offset path separation

Offset	Center	Left	Right
0	Box center on master wire centerline	Left edge master wire coincide with right edge of box	Right edge master wire coincide with left edge of box
>0	Shift left (separation value is distance between box center and master wire centerline)	Shift left (separation value is distance between left edge of master wire and right edge of box)	Shift right (separation value is distance between right edge of master wire and left edge of box)
<0	Shift right (separation value is distance between box center and master wire centerline)	Shift right (separation value is distance between left edge of master wire and right edge of box)	Shift left (separation value is distance between right edge of master wire and left edge of box)

5.7: Contact

The **Edit→Create Object→Contact** menu option activates the contact generating tool. The tool has two modes:

- **Automatic**, in which it selects parameters for the new contact depending on the parameters of overlapping wires. In this mode, contact could be placed only at places where two or more wires overlap.

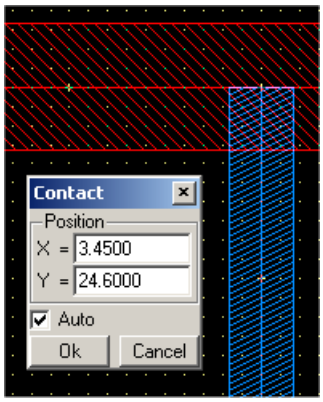


Figure 5-23: Placing the Contact in Automatic Mode

- **Manual**, in which you can define parameters for new contacts and place them wherever you want.

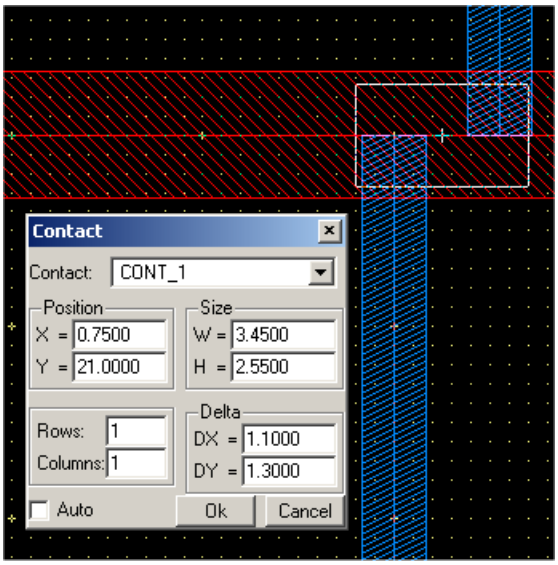


Figure 5-24: Placing the Contact in Manual Mode.

In Automatic mode, simply point the cursor where wires overlap and click left mouse button, and the appropriate contact will be created.

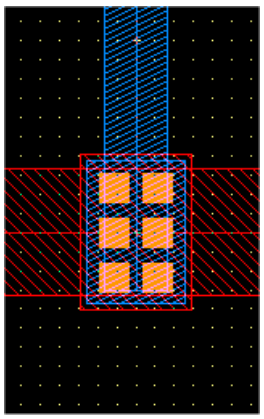


Figure 5-25: Example of a Contact

In Manual mode, you can select a contact template from the drop down list of contacts defined in the technology. When you select a template, all other fields in the Numeric input are filled with predefined values from the technology. You can override all these values before placing the contact. The **Width** and **Height** fields define the size of the box object that is created in the contact layer. This size does not include margins in contacted layers, which are defined in technology, so the actual size of contact cell could be different. Usually contact dimensions are equal to the width of wires, of which the contact is created, but the contact cell might contain several rows and columns of box objects in the contact layer. To create such a cell, you define how many rows and columns you need, and what the gap between rows and columns will be. When you entered all parameters, place the mouse cursor wherever you want the contact to be placed and click left mouse button. Alternatively, you can enter the coordinates of the new contact in the Numeric input panel and press **OK** to create contact.

5.8: Bus

Use the **Bus** tool to route several wires simultaneously. Essentially, it works just like the **Wire** tool, except instead of single wires, multiple parallel wires are created.

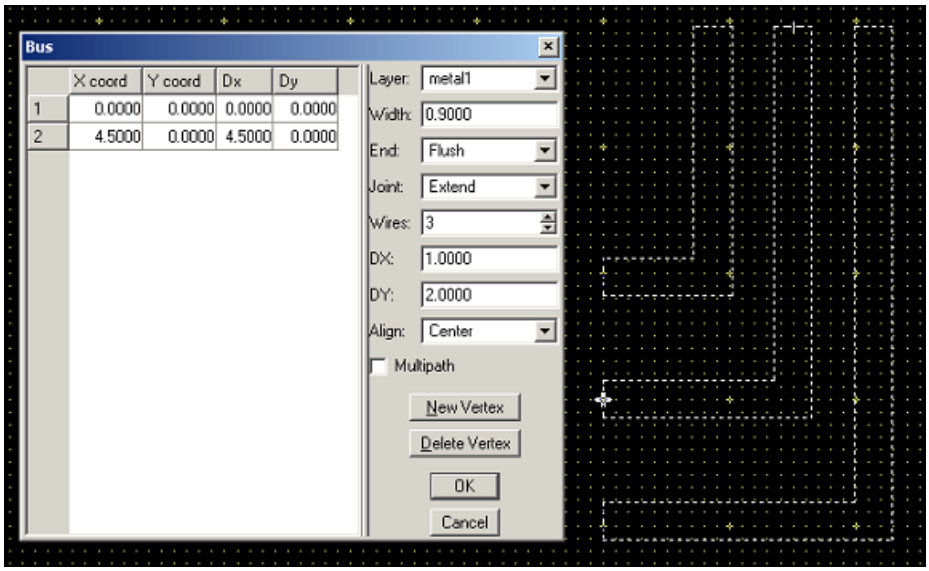


Figure 5-26: Bus

Once you selected the menu command **Edit→Create Object→Bus**, the Numeric input panel pops up. This form is similar to the ordinary wire's Numeric input, but it allows you to specify few additional parameters:

- Number of wires that will be routed.
- Horizontal and vertical spacing between wires.
- Alignment of wires, or how wires lie relatively to line that defines path.

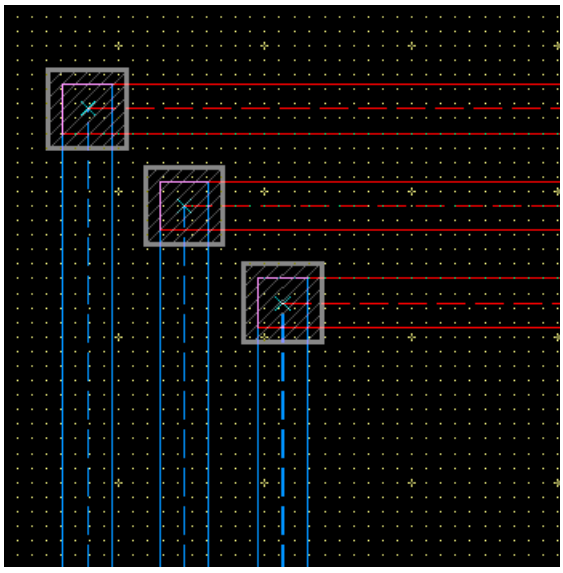


Figure 5-27: Bus Routed Through Two Layers

Just as with the regular wire, Bus could be routed through multiple layers. The procedure is the same:

- Make sure contacts are properly defined in the technology.
- Select a layer where the route will start.
- Checkmark the Multipath checkbox.
- Start routing bus.
- Select different layers in the **Layer** combobox.
- Continue routing the switching layers as needed.

Chapter 6:
Modifying Layout Objects

6.1: Objects

To modify the shape of a single existing object, select **Edit→Modify Object**. The **Modify Object** option changes an object by direct manipulations with components of its shape (such as vertices and sides). There are, however, some more complicated shape modifying commands as described in other chapters.

- Stretch
- Jog Points
- Cut by Line
- Clip out / Clip In
- Merge Selected
- Resize Selected
- Scaling
- Snapping to grid
- **Merge** mode of **Create Object** commands
- **Hole** mode of **Create Object** commands
- Running xi-scripts
- Execution of DRC commands and running DRC scripts

In addition, you can change the object's attributes and user-defined properties in the **Property Bar** or in the **Numeric Input** panel.

Finally, many common editing operations that do not change the shape of primitive objects are described in Chapter 8: "Other Edit Submenu Operations".

6.2: General

The **Modify** tool can change only selected objects. This way you know exactly what object you are modifying. If several objects lie close to each other, or even overlap, then you can select the object you want to modify and perform transformations only on that object (other objects won't be touched).

To use the **Modify** tool, either select **Edit→Modify Object** from the menu or press the corresponding button from the **Design Bar** (see Figure 6-1).

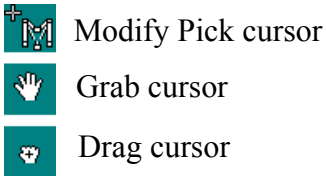


Figure 6-1: Modify objects

The cursor assumes the **Modify Pick** shape. It indicates that you can pick an object for modification by clicking on it. In this mode, the **Modify** tool behaves in the same way as **Select By Click/Drag** tool (see Chapter 7: "Selecting Layout Objects").

If you move the cursor over the selected object, it changes its shape to the **Grab** cursor. It indicates that you can grab object's segment, corner or even the whole object (it depends which part of the object cursor is moved). If pre-selection is on, you can actually see what part of the object would be grabbed in each particular point of time.

In order to modify the object, grab the object or its part, and move it on desired offset. You can grab and move something with the **Modify** tool by either pressing the mouse button, dragging the mouse, releasing the mouse button, or by clicking the mouse button, moving mouse, and clicking mouse button again.

When you are dragging segments of an object, they remain parallel to themselves. For adjacent segments, two modes are available (see Figure 6-2).

- If you check **Edit→Angle Mode→Preserve Angles Mode**, the ends of the picked side slide along the two neighboring sides so that its length varies while its adjacent angles do not. This mode always works for **45/90 degree mode**.
- Otherwise, the picked side translates freely so that its length does not change, while the two adjacent sides stretch/contract freely. This mode only works for **All angle mode**.

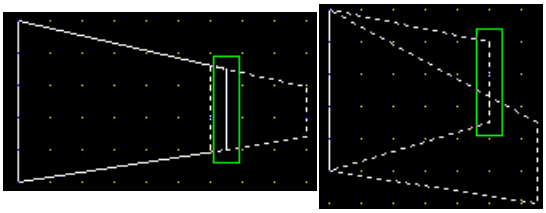


Figure 6-2: Side Drag Modes

Some object's attributes cannot be changed using the mouse (for example Layer). For such attributes you should use the **Property Bar** or **Numeric Input**. Attributes of different types of objects were described in the Chapter 5: "Creating Layout Objects".

You can use the **Modify** tool in combination with other tools. For example, if you want to add a new vertex to a polygon, or new joint to wire, you can use **Add Jog Points** tool. For more information about these tools, see Chapter 8: "Other Edit Submenu Operations".

Chapter 7:
Selecting Layout Objects

7.1: Layout Objects

You may often need to copy, move, delete or do something else with several layout objects. In this case, you must mark (select) objects to be processed simultaneously. The selected objects are highlighted in the layout. This chapter describes operations that permit you to select/deselect (sets of) whole objects in the edited cell. To invoke these operations, use the **Select** submenu (see Figure 7-1). Several settings from this submenu affect the results of selection or facilitate the selection process.

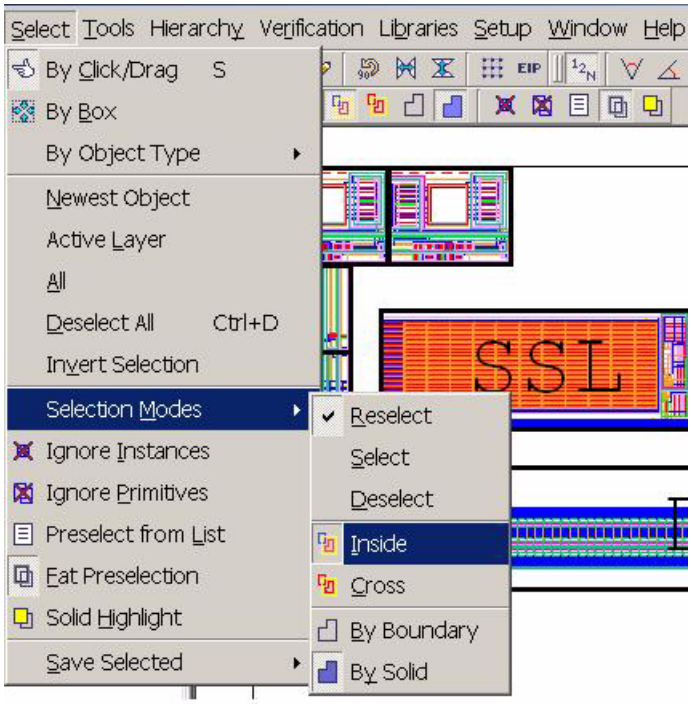


Figure 7-1: Selection Menu

Some operations (Modify, Stretch) require partial selection of shapes, in other words, separate vertices or segments or both. This kind of selection is described together with the corresponding commands. To avoid confusion, all other kinds of selection are referred to as **picking**.

Selection is probably the most frequent editor operation. Therefore, **Setup→Editor/ Viewer...→Editing** has an option: **Selection tool as default**. If this option is on, then every time you exit from any other operation using **Esc** key, you switch to the **Selection** tool. Otherwise, you would switch to the **Void** tool.

Most frequent selection commands and options are placed into the **Selection Bar**. Recall that you can show/hide control bars by selecting **View→Control Bars...**.

7.2: Select By Click/Drag

Action

1. **Select→By Click/Drag** selects a layout object if you point to it and click the left mouse button. EXPERT's message line will then contain information about the selected object.
2. Selects layout objects by box by dragging a mouse.

Execution

This action is invoked from the Select menu or by the corresponding button from the **Design bar**. You then select objects several times until you switch to another command (or to the **Void** tool).

There are several modes of selection described in Section 7.2.1: "Selection Modes".

Nonlinked texts are selected if you click anywhere inside its frame. A linked text cannot be selected separately from its linking object. If you click on the linked text, you will select its linking object as well.

If **Select→Preselection** mode (described below) is off, then the exact position of the cursor to select a primitive object depends on the current selection mode: **boundary** or **solid**.

- If **Select→Selection Modes→By Boundary** mode is on, you must point at the border line of the required object.
- If **Select→Selection Modes→By Solid** mode is on, objects are (de)selectable by clicking on any of its inside or boundary points.

The borders of the selected primitive objects are highlighted. If the menu option **Select→Solid Highlight** is checked, then the interiors of the selected primitive objects are highlighted.

Note: If you are in **Select by Box** operation (see below), double-clicking on an object emulates pointer selection.

Note: If you are in the **Select→By Click/Drag** operation mode, you can select by box as well. To draw a selection box, press the left mouse button at the first corner of the desired selection box, drag the cursor into the opposite corner, and then release the button.

7.2.1: Selection Modes

Select→Selection Modes→Select: In this mode, the picked objects are added to the selected set.

Select→Selection Modes→Reselect: In this mode, the previously selected objects are deselected and the picked objects become selected.

Select→Selection Modes→Deselect: In this mode, the picked objects are removed from the selected set.

The mode set from **Select** submenu is called **active** here.

Shift key depressed: Toggles between the Reselect or Deselect active mode and Select mode.

Ctrl key depressed: Toggles between the Reselect or Select active mode and Deselect mode.

7.2.2: Selection Among Overlapping Objects

If the cursor points at several overlapping objects, you can choose which of them you need to select/deselect in several ways.

- Use **List Selection** mode described in Section 7.4: "List Selection".
- Hold the mouse still at the same point and click the left mouse button several times. You will see how your selection changes for the objects which are under the cursor. Notice also that the preselection highlights the next object to be selected if you click next time.
- Hold down the **Alt** key and click the left mouse button several times. You will see that your selection acts for the objects, which are under the cursor does not change but the preselection changes. If you release the **Alt** key and click once more (with mouse still), then the preselected object will be selected.

If you think that the above is too tricky, the **Preselection mode** described below makes this task easier.

Note: In order to reduce the choice, you may hide some layers or make them non-selectable

---

7.2.3: Preselection Mode

Preselection mode greatly simplifies selection among overlapping objects.

The **Select→Preselection** menu mode acts for various single-object picking selection tools. For example, single-object selection, picking for the Modify operation, entering into Edit in place, or for node probing.

Here, we describe the action of this mode for the selection tool (**Select→By Click/Drag** menu command).

If this mode is on, then the object with a boundary sufficiently close to the cursor position is highlighted.

If you click the left mouse button, the highlighted object is selected. In order to be preselected, the object's boundary must be closer than 10 pixels to the cursor. If there are several such objects, then the closest one is preselected.

Note: Ellipces/circles can be displayed either as a true ellipse/circle, or as expected polygonal shapes, depending on the options in **Setup→Editor/Viewer...→Viewing**. The preselection marker for ellipses/circles shows the shape that will be obtained if the ellipse/circle is converted to polygon with the current settings (vertex count, fitting to grid, etc.).

Note: The **Select→Fat Preselection** mode make the appearance of the preselected object more visible.

Note: If several objects are laying under the current cursor position, holdingthe **ALT** key will help choose the desired object.

7.3: Box Selection

Purpose

Selection/deselection of layout objects that are either completely inside or intersect the selection box, depending on **Inside** or **Cross** settings in the **Select→Selection Modes** submenu.

Execution

This action is initiated in two ways.

1. By the **Select→By Box** command, or by the corresponding button from the **Design bar**. Once you have invoked this command, you may perform box selection several times until you switch to another command (or to the **Void** tool). The selection box is specified by clicking the mouse at any pair of its opposite corners.
2. If you are in the **Select→By Click/Drag** operation, you may select by box as well. The selection box is drawn by a press-drag-release action: You must press the left mouse button at the first corner of the required selection box, drag the cursor into the opposite corner, and then release the button.

The set of objects to be affected by box selection depends on the current setting of **Selection Mode**:

- If **Select→Selection Modes→Inside** mode is on, the objects that fit completely inside the selection box changes their selection state.
- If **Select→Selection Modes→Cross** mode is on, the objects that fit completely inside the selection box or intersect its boundary changes their selection state.

The **Select/Reselect/Deselect** modes act as those described above.

Figure 7-2 demonstrates the difference between the **Inside** and **Cross** selection modes. The first figure shows the initial state of objects and the selection box. The second figure shows the selection made in **Inside** mode. The third figure shows the selection made in **Cross** mode.

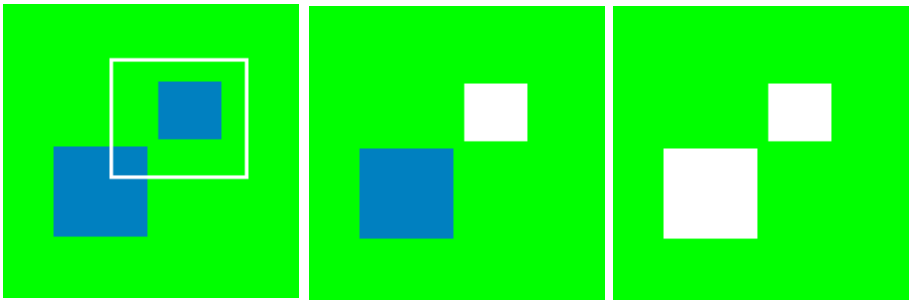


Figure 7-2: Selection modes: inside and cross

7.4: List Selection

It is possible to obtain the list of objects which are under the cursor or hit the selection box. You can preview them one by one and select the required ones.

Check **Select→Preselect from list** option, and start selecting by left mouse click or by box (in either way: by two clicks or by drag).

You will see the list of all objects that may be selected in **Preselected Objects List** dialog, as shown in Figure 7-3.

This list has two columns:

- **Type** shows object type (box, wire, cell instance, etc.).
- **Layer/Cell** shows the name of the corresponding layer or cell.

If you click once or twice at the title of a column, the lines will be sorted by this column.

If you pick some objects in the **Object** column, then the corresponding objects in the layout will be highlighted. The **Property Bar** shows info for the highlighted objects.

If you click the **Select highlighted objects** button in **Preselected Objects List** dialog, highlighted objects will become selected. **Select all found objects** button makes all listed objects selected.

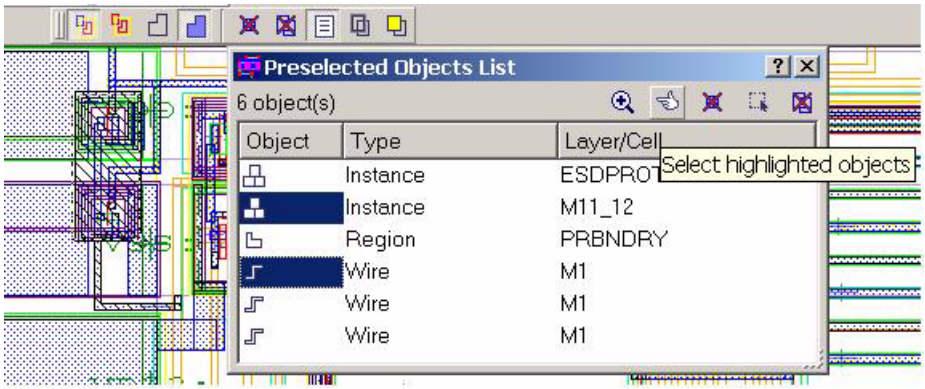


Figure 7-3: Preselection from List of Objects

7.5: Deselection

Purpose

Cancels selection state for objects in the current cell. This operation may be invoked in various ways.

Execution

1. If selection by the mouse pointer and selection by box (see previous sections this chapter) is used, one must to switch to **Select→Selection Modes→Deselect** menu mode. as it was described in Sections 7.2: “Select By Click/Drag” and 7.3: “Box Selection”. In particular, if you are in Select of Reselect mode you may depress the **Ctrl** key and pick object for deselection.
2. The **Invert Selection** command of the **Select** submenu may be used. In this case, all selected objects become deselected and vice versa.
3. The **Deselect All** command of the **Select** submenu cancels the selected state of all objects of the current cell.
4. Finally, while in **Reselection** mode, a left mouse click in the empty space of the layout, will deselect all objects in the cell.

7.6: Save Selected As Cell

Purpose

Save Selected→As Cell creates a new cell from the set of selected layout objects and add it to the project. This operation is the same as the **Cell→Create...** menu command.

Execution

- Select a set of layout objects which you want to lump together into a new cell.
- Click on the menu command, **Select→Save Selected→As Cell...**. The Cell Name Selection Form with the title "**Create Cell**" and the following fields appear.

New Cell Name: You must type a name for the new cell into this field. Initially this field contains a proposed default name. It will be used if you do not enter another name.

List of Cells: This is the list of all cell names existing in the project. If you click on a name from this list, it appears in the **New Cell Name** field and you may further edit it here to produce a new name.

Cancel: Pressing this button abandons this operation.

OK: Pressing this button, after selecting the cell name, completes the operation.

If the **OK** button was pressed, two outcomes are possible:

- A message box "The project already has such a cell name!" appears. You must click the **OK** button of the message box and select another name.
- **Create Cell** dialog box closes. The operation was performed successfully.

Results

This operation does not change the active cell; it only adds a new top-level cell to the project.

If the operation is performed successfully, the project changes as follows:

- A new cell is added to the project. It consists of the copies of the selected objects of the active cell. The reference point of the new cell is at the lower left corner of the bounding box of the selected objects, and the local coordinates of the copied objects in the new cell are obtained by subtraction of the bounding box's lower left corner coordinates from the selected object's coordinates in the active cell.
- The active cell remains unchanged.
- The selected objects remain selected in the active cell.

See also: **Hierarchy →Create Cell in-Place, Cell → Create...**

7.7: Save Selected Copy To Active Layer

Purpose

Save Selected→Copy To Active Layer menu item copies the set of selected layout objects into the current layer.

Execution

- Select a set of layout objects to copy into the current layer. These objects may be from different layers.
- Click the **Select→Save Selected→Copy To Active Layer** menu item.

Results

The selected primitive objects and texts will be copied into the current layer. This operation does not involve selected instances and arrays (it is meaningless for them).

7.8: Save Selected Move To Layer

Purpose

Save Selected→Move To Layer menu item moves the set of selected layout objects into the current layer.

Execution

- Select a set of layout objects which you want to move into the current layer. These objects may be from different layers.
- Click the **Select→Save Selected→Move To Layer** menu item.

Result

The selected primitive objects and texts will be moved into the current layer. This operation does not involve selected instances and arrays (it is meaningless for them).

7.9: Other Selection Commands/Options

7.9.1: All

The menu command, **Select→ All**, selects all visible and selectable objects in the current cell.

7.9.2: Active Layer

The menu command, **Select→Active Layer**, selects all objects in the active layer of the current cell. The active layer must be selectable, see Chapter 11: “Setup,”, Section 11.4.3: “Layer/Wire Setup”.

7.9.3: Selection by Object Type

The **Select→By Object Type** command allows to choose objects among the following types:

- Cell Instances/Arrays
- Cell Arrays
- Texts
- Shapes
- Boxes
- Regions
- Wires
- Circles
- Ellipses

When the **Cell Instances/Arrays...** is chosen, the cell selection dialog box appears with the list of cells instanced in the current cell. Several cells may be marked from this list, and all instances of all marked cells will be selected in the edited cell.

7.9.4: Select Newest

The menu command, **Select→Newest Object**, selects the newest object, in other words, the one resulting from the most recent **Create** or **Modify** command in the edited cell.

Note: Option **Setup→Editing→Select newest object** allows you to automatically select the newest created or modified object.

7.9.5: Invert Selection

The menu command, **Select→ Invert Selection**, inverts the selection state for all objects of the current cell. All selected objects become unselected and vice versa. As usual, it acts only on visible and selectable objects.

7.9.6: Cell Instance Selectivity

Select→Ignore Instances command allows you to switch off/on the selectivity of cell instances in the layout.

7.9.7: Primitive Object Selectivity

Select→Ignore Primitives command allows you to switch off/on the selectivity of primitive objects in the layout.

7.9.8: Selection Count Indicator

At the right end of the message line (bottom of EXPERT'S window) there is an indicator showing the total number of currently selected objects in the edited cell. It looks like this: "Sel: 5/2", where the first number is the number of all selected objects and the second one is the number of selected instances. If you need more detailed information about selected objects, use the **Cell→Info...** menu command.

More information about selected objects may be obtained by the menu command, **Cell→Selection Info...**. The Info window is resizable to see more data without scrolling.

7.9.9: Selection Highlight Modes

The following mode acts only for a Solid or a Stipple kind of filling. If the menu item, **Select→Solid Highlight** is checked, the whole interior of a selected object is highlighted white. Otherwise, only the object's border is highlighted.

7.9.10: Select Newest Objects

If the **Setup→Editor/Viewer..→Editing→Select Newest Objects** option is on, the most recently created or modified objects are selected in accordance with the actual Select/Reselect mode.

7.9.11: Selection Tool as Default

If the **SetupSetup→Editor/Viewer..→Editing→Editor/Viewer..→Editing→Selection Tool as Default** mode is on, then whenever you exit from any other editing tool, you are switched to the **Select by Click/Drag** tool.

7.9.12: Fat Preselection

The **Select→Fat Preselection** mode changes the appearance of the preselection marker.

Sometimes the selected edges for stretch operation are not clearly visible. If this mode is on, then selection for stretch is by fat lines as well.

7.9.13: Layer Selectability/Visibility

Primitive objects may be selected only in layers that are both visible and selectable, see Chapter 11: "Setup,", Section 11.4.3: "Layer/Wire Setup".

7.9.14: Setup Hotkeys/Menus/Colors... Color

See Chapter 11: "Setup," for description how to select the color for selection and preselection markers.

Chapter 8:
Other Edit Submenu Operations

8.1: Editing Commands

This chapter describes EXPERT commands from the **Edit** submenu, with the exception of **Create** and **Modify** operations, described in Chapters 5: "Creating Layout Objects" and 6: "Modifying Layout Objects" respectively.

This submenu contains commands to perform simple editing of the layout in terms of **selected** (highlighted) objects:

- cut/copy/paste using clipboard
- geometrical transformations

as well as:

- access to user-defined object properties
- search according to various criteria
- stretch operation, etc.

The **Tools** submenu of EXPERT contains still other complex editing operations, described in Chapter 12: "Tools".

Note: The shape of the cursor is usually indicative of the current editing operation of EXPERT.

Note: When EXPERT is not running editing operations, it is said that the **Void Tool** is running. It is advisable to set this mode to prevent inadvertent editing during browsing of the layout. Set EXPERT in this mode by using the **Tools→Void Tool** menu command.

Note: When you exit from any editing mode, e.g., by hitting the **<Esc>** key, the editing state of EXPERT is switched to the **Default tool**. The latter one can be either the **Void tool** or the **Select tool**. The choice depends on the **Selection tool as default** setting in the **Setup→Editor/Viewer...→Editing** form; see Chapter 11: "Setup".

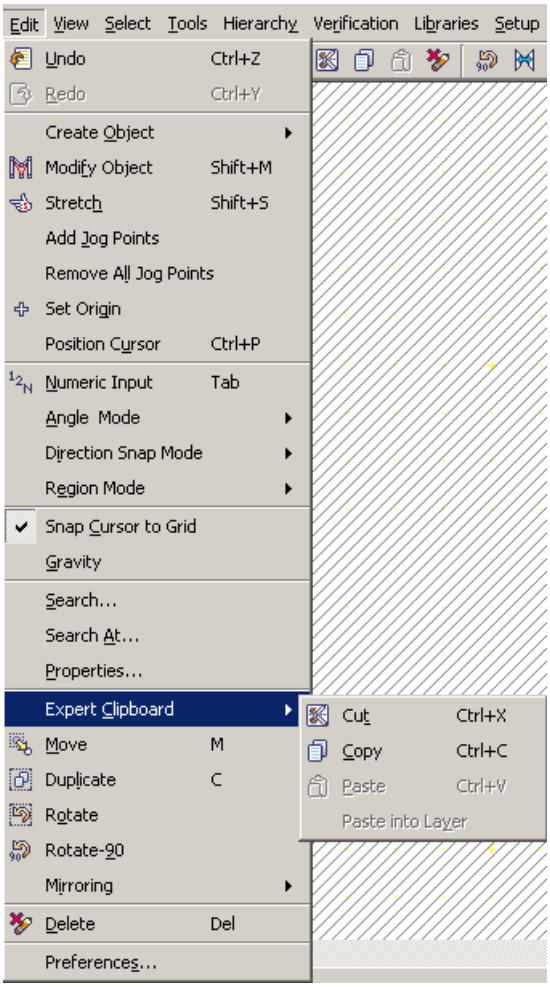


Figure 8-1: Edit Submenu

8.2: Expert Clipboard

The EXPERT **Clipboard** is the editing buffer designed for temporary storage of layout objects. It may be used in the same way as the **Windows** clipboard. You can cut or copy objects into it and then paste them into the required position of the current cell, as well as into other cells.

Note: EXPERT clipboard is not the same as the Windows clipboard. In particular, you cannot paste data from EXPERT's clipboard into other Windows applications and vice versa.

8.2.1: Cut

Purpose

The **Edit→Expert Clipboard→Cut** menu option transfers the selected objects of the current cell onto the clipboard. Every object pasted onto the clipboard is accompanied by information about its layer and position relative to the group frame, where the group frame is the minimal rectangle enclosing the selected objects. The **Cut** operation replaces the previous contents of the EXPERT clipboard by the currently selected objects. This operation may also be invoked from the **Command** bar.

8.2.2: Copy

Purpose

The **Edit→Expert Clipboard→Copy** menu option copies the selected objects of the current cell onto the EXPERT clipboard. Every copied object is accompanied by information about its layer and position relative to the group frame, where group frame is the minimal rectangle enclosing the selected objects. The **Copy** operation replaces the previous contents of the EXPERT clipboard by the currently selected objects. This operation may also be invoked from the **Command** bar.

See also **Edit→Duplicate** command for copying data within the same window.

8.2.3: Paste

Purpose

The **Edit→Expert Clipboard→Paste** menu option inserts the set of objects contained on the EXPERT clipboard into the current cell. This operation may also be invoked from the **Command** bar. To insert these objects into another cell, it must be opened and made current before invoking the **Paste** operation.

EXPERT prompts you to point at the position where the object will be inserted. To this end, the bounding box of the objects to be pasted is attached to the cursor. You can then drag this box over the screen to select a proper position for pasting. At the selected place, click the left mouse button to drop the objects here. The relative positions of objects, as well as their layers, will be preserved.

Figure 8-2 demonstrates the **Paste** operation. The first picture shows three layout objects, two of them being selected. The white rectangle near the cursor at the bottom of the second picture is the box illustrating the required location of the inserted objects. The two rectangles at the bottom of the third picture are the objects from the clipboard inserted at the position of the bounding box.

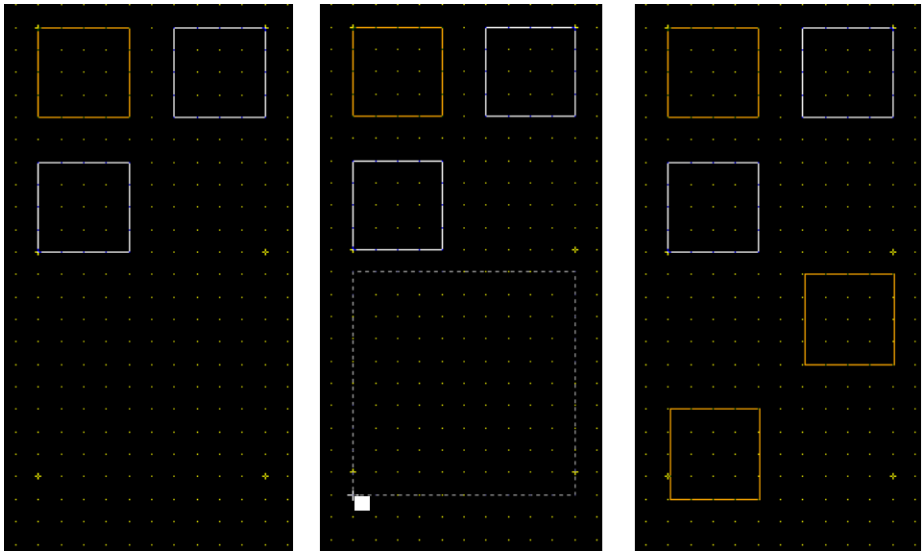


Figure 8-2: Paste operation

8.2.4: Paste into Layer

Purpose

The **Edit→Expert Clipboard→Paste into Layer** menu option inserts the set of objects contained on the EXPERT clipboard into the current layer of the current cell. The **Paste into Layer** operation works like the **Paste** operation with two natural exceptions:

- This operation ignores the selected instances/arrays.
- During the **Paste into Layer** operation, all objects are inserted into the current layer, regardless of their original layers.

8.3: Delete

Purpose

The **Edit→Delete** menu option deletes the set of selected objects of the current cell from the layout. This operation may also be invoked from the **Command** bar. It does not change the contents of the EXPERT clipboard.

8.4: Position Cursor

The **Edit→Position Cursor** menu option allows you to position the cursor at given coordinates. By this command you must input the required coordinates at the metric bar, and then press the **Enter** key. The cursor positions itself at the required coordinates. If necessary, the screen pans to this point.

Note: Screen resolution must be sufficient for unambiguous transformation of the coordinates on chip into screen pixels; otherwise, the positioning will be approximate.

Note: Use **Tab**, **Shift-Tab** to move the cursor from one coordinate to another.

8.5: Geometric Transformations

8.5.1: Edit Duplicate, Edit Move

The two commands **Edit→Duplicate Edit→Move** are very similar in their user interface. **Move** will change the selected object's position; **Duplicate** will create new objects.

After beginning to move objects, you can rotate or flip objects. Pressing the right mouse button will rotate object counter-clockwise. Holding the **Shift** button and pressing the right mouse button will flip objects.

Duplicate

Purpose

Copies the set of selected objects of the current cell to another position in the cell, shifted with respect to their initial positions by a given vector. This operation may be invoked from the menu and the **Design** bar. It does not change the contents of the EXPERT clipboard.

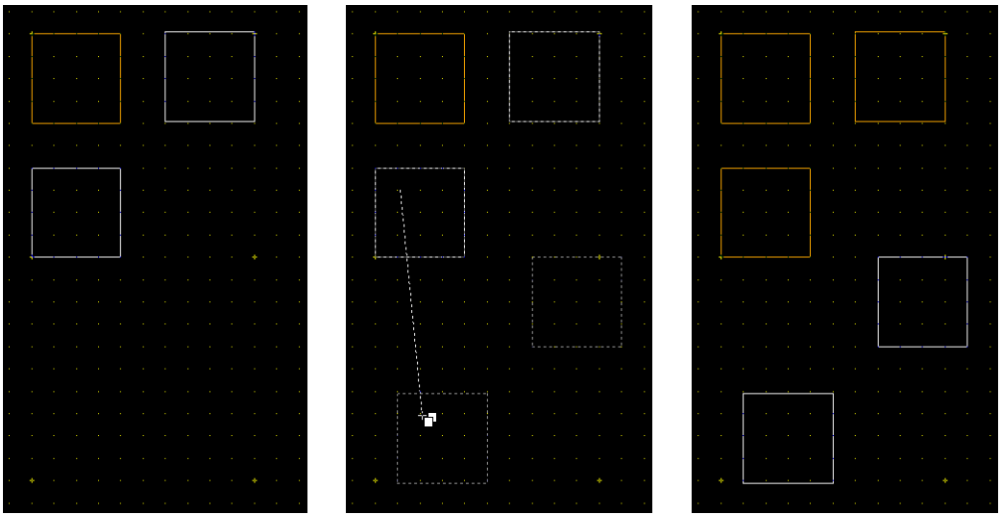


Figure 8-3: Duplicate Operation

Execution

This operation is similar in its execution to the **Edit→Move** operation, described in "Move" on page 8-6. After the selection of this operation either from the menu or from the **Design** bar, you must define the shift vector. To do this, click first at the tail and then at the head of the required shift vector. The vector may be chosen anywhere in the layout at your convenience. When the shift vector head is clicked, the new objects are created, and shifted by this vector with respect to the initial (selected) objects. The new objects belong to the current cell; their relative positions and their layers are the same as for the initial objects.

Note: After the duplicate operation, the originals are deselected and the copies become selected.

Figure 8-3 demonstrates the stages of the **Duplicate** operation. The first picture shows three layout objects. Two selected (highlighted) objects are to be duplicated into a new location. The white dash line at the second picture shows the shift vector. Two highlighted (selected) boxes at the bottom of the third picture are the new objects created after clicking the mouse button at the head of the dash line.

Property Bar Mode or Numeric Input Mode

If the **Numeric Input** panel is visible, once you have selected the item(s) to duplicate, the **Numeric Input** displays delta dx and dy of the shift vector. These values can be typed in directly to input the coordinates of the shift vector. Pressing **Enter** or clicking on the **OK** button will perform the actual copying.

Move

Purpose

Moves the set of selected objects of the current cell to another position in the current cell, shifted relatively to their initial position by a given vector. This operation may also be invoked from the **Design** bar. It does not change the contents of the EXPERT clipboard and the selection state of the objects.

Execution

It is similar to that of the Duplicate operation. After selecting this operation either from the menu or from the **Design** bar, you must define the shift vector. To do this, first click on the tail, and then on the head of the required shift vector. The vector may be chosen anywhere in the layout at your convenience. When the shift vector head is clicked, the selected objects are shifted by this vector with respect to their initial positions. Figure 8-4 demonstrates the stages of the **Move** operation. The first picture shows three layout objects. Two selected (highlighted) objects are to be moved into a new location. The white dash line in the second figure shows the shift vector. The third figure shows the layout created after clicking the mouse button at the head of the dash line.

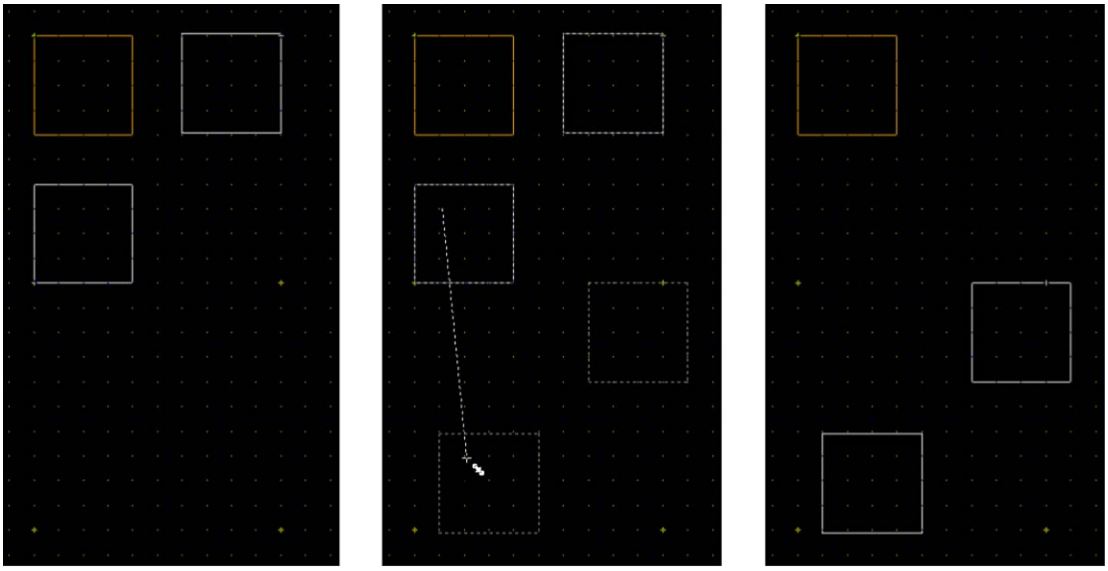


Figure 8-4: Move Operation

Property Bar Mode or Numeric Input Mode

If the **Numeric Input** panel is visible, once you have selected the item(s) to move, the **Numeric Input** displays delta dx and dy of the shift vector. These values can be typed in directly to input the coordinates of the shift vector. Pressing **Enter** or clicking on the **OK** button will perform the actual copying.

8.5.2: Rotate-90

Purpose

The **Edit→Rotate-90** menu option will rotate the set of selected objects of the current cell by 90 degrees counterclockwise around the center of the minimal bounding box of the selected objects. In addition, the **Setup→Editor/Viewer...Editing** form provides a self-descriptive option **Flip/rotate single selected instance/array with respect to its origin**.

The **Rotate-90** operation can be launched from the **Edit→Rotate-90** menu of from the **Command** bar icon.

The operation does not change the contents of the EXPERT clipboard and the selection state of the objects.

8.5.3: Rotate

Purpose

The **Edit→Rotate** menu option will rotate the set of selected objects of the current cell by any angle with respect to any point, see Figure 8-5.

Execution

There are three ways to perform this operation: by mouse actions, using numeric input dialog, or by xi-command. See Chapter 17: “Expert Interface Scripts (XI-Shell)” for the xi-script syntax of this command.

Rotation by Mouse

- Click the left mouse button to define the center of rotation;
- If you move the mouse slightly, you will see the following:
  - the **outline** of the selected objects rotated with respect to the initial position;
  - the **vector** starting at the rotation center and ending at the current cursor position.

The vector defines the current angle of rotation with respect to the horizontal direction. This angle is also shown in the **Tool: Rotate** tab of the **Property** bar, in the “Rotation” field.

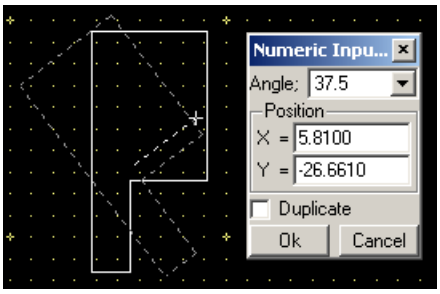


Figure 8-5: Rotation by Any Angle

- The second left mouse click defines the required angle of rotation.

Property Bar Mode or Numeric Input Mode

If the **Numeric Input** panel is visible, once you have selected the item(s) to rotate, the **Numeric Input** displays a form which includes the Rotation angle. This value may be typed in to input the required parameters of rotation, shown in Figure 8-5.

The Duplicate option in the **Numeric Input** panel of the Rotate command acts as follows.

- If this option is “true”, the Rotate command copies the set of selected objects.

- If this option is "false", the Rotate command moves the set of selected objects.

Note: 1. You may combine mouse clicks and **Numeric Input**. For example, you may click the center of the rotation by mouse, then hit the **Tab** key to jump into the **Numeric Input** dialog and input the numeric value for the rotation angle.

Note: 2. Rotation of cell arrays by a non-orthogonal angle is disallowed.

8.5.4: Mirroring Operations

Purpose

To perform mirror reflection of the selected objects in the edited cell. There are two simplified mirroring operations: vertical and horizontal flip.

Edit→Mirroring→Vertical Flip operation reflects the objects with respect to the horizontal axis. **Edit→Mirroring→Horizontal Flip** reflects the objects with respect to the vertical axis.

By default, the mirroring axis goes through the center of the minimal bounding box of the selected objects. In addition, the **Setup→Editor/Viewer...→Editing** form provides a self-descriptive option, **Flip/rotate single selected instance/array with respect to its origin**.

Figure 8-6 shows the two selected (highlighted) objects before and after mirroring. The illustration in the center shows a Vertical flip of the left illustration. The Horizontal flip is presented in the illustration to the right. The operation does not change the contents of the EXPERT clipboard and the selected state of objects.

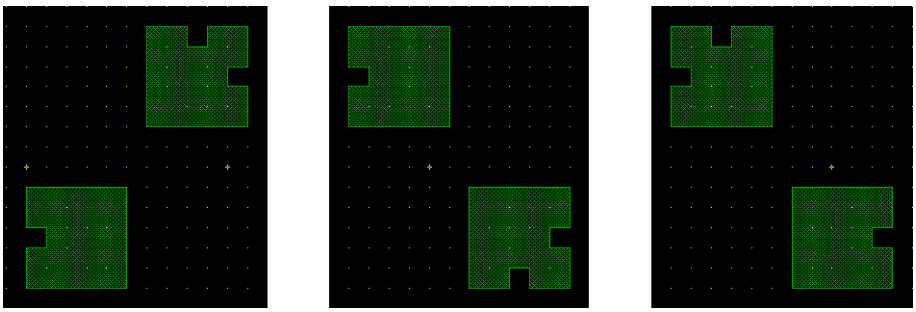


Figure 8-6: Horizontal and vertical flips

8.5.5: Set Origin

The menu command, **Edit→Set Origin**, allows you to select a new origin for the current cell by clicking on the mouse at the required point.

8.6: Undo/Redo

An Undo operation of unlimited rollback depth is implemented with its own undo list. The Undo is invoked in one of the following three ways:

- by menu command **Edit→Undo**,
- by clicking the **Undo** button at the Command bar, or
- by pressing **Ctrl-Z**.

The **Esc** key, an additional option, interrupts editing.

In general, undo is applicable only to editing operations. You cannot undo pan/zoom, select operations, switching among windows, etc.

If the undo list is empty, the undo button is gray and inactive.

Note: Some operations **are not** undoable if the numbers of changed objects exceeds the settings **Maximum size of Undo step [N] objects** (see Section 11.3.4: "Data Safety Page"). You will get a warning in this case.

8.6.1: Esc Key Interrupts Editing Operations

The **Esc** key interrupts many operations of EXPERT, such as loading.

In particular, if you start creating an object, pressing the **Esc** key deletes the unfinished object. In other words, it serves as a limited undo operation.

8.6.2: Undo

The Undo operation is available outside of any editing operation that allows you to roll back along the completed editing operations.

Note: 1. Be careful during a long roll back: you may undo changes you do not see.

Note: 2. If a selected object undergoes an undo operation, it remains selected.

8.6.3: Redo

A **Redo** operation reverses the results of the undo operation.

8.7: Stretch Operation

8.7.1: Overview

This operation allows you to modify several sides of several regions and several wires at the same time.

Most time Stretch operation will consist of two steps:

- Picking several region sides and/or several wire segments, possibly from different regions, boxes, and wires. They will move during the stretch;
- Specifying the amount of the stretch (stretching shift). All picked parts of objects will be moved/stretched on that vector.

Sometimes you will also need to change some options which affect how Stretch works. This option will be described further.

During the stretch, the segments will be classified as **movable**, **fixed** or **stretchable**:

- The picked segments will be moved as a whole, so they are called **movable**.
- The segments with both ends shared with some picked segments will move as a whole, so they are **movable** as well.
- The segments with no ends shared with movable segments will rest in place, so they are **fixed**.
- The segments with one end shared with a movable segment, and the other one by a fixed segment will stretch to accommodate the shift of the picked segments, so they are **stretchable**.

These segment types may also be seen in Figure 8-7. The left sides of boxes are fixed, right sides are movable, and the horizontal sides are stretchable.

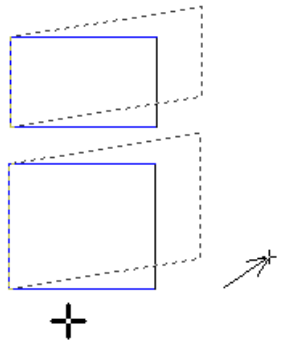


Figure 8-7: Stretch operation

During cursor movement, you will see preselected object segments. If you will click the left mouse button at this position, a preselected segment is picked for stretch.

If the cursor points at several overlapping segments, you can choose which of them you need to select/deselect. Hold down the **Alt** key and click the left mouse button several times. Your preselection will change cyclically among the overlapped segments. If you release the **Alt** key and click once more (with mouse still), then the preselected segment will be selected.

8.7.2: Sample Run

The **Stretch** operation is controlled by several settings. To get a glimpse on how it works let's step through the following scenario:

1. Draw a box.
2. Ensure that following options are set:
   - **Edit→Angle Mode→Preserve Angles Mode** is turned off.
   - **Edit→Direction Snap Mode** is set to **None**.
3. Pick one side of the box by clicking on it. One edge of the box should become selected.
4. Grab the selected part by pressing the left mouse button and moving it around while holding mouse button. You will see how the box is stretched.
5. Turn on **Edit→Angle Mode→Preserve Angles Mode** option.
6. Pick two adjacent edges of the box by clicking on one of its corners. These edges should become selected.
7. Grab the selected part and move it around by holding mouse button. You will see another sort of stretching.

8.7.3: Specification

The behavior of the Stretch operation is controlled by the following modes:

- set selection options,
- enter the Stretch operation in mouse mode,
- specify the stretch shift value by numeric input.

In addition, the behavior of the Stretch operation is controlled by the following modes:

- **Edit→Direction Snap Mode**;
- **Edit→Angle Mode→Preserve Angles Mode**;
- **Select→Selection Modes**.

Selection (Pick) Modes for Movable Segments

When you are picking/selecting something with the **Stretch** tool, it works similar to **Select By Click/ Drag** tool described in the Chapter 7. The difference is that **Stretch** tool can select individual parts of objects, but both tools are controlled by the options grouped in the **Select→Selection Modes** menu. For explanation of options see Chapter 7: "Selecting Layout Objects".

You can select/deselect segments of regions/wires for stretching by clicking on them. Alternatively, you can drag the selection frame and select parts of objects in that way.

Stretching Modes

The stretch itself is controlled by **Edit→Angle Mode→Preserve Angles Mode** and **Edit→Direction Snap Mode**.

The **Preserve Angles Mode** acts exactly in the same way as for the single-object **Edit→Modify Object** operation.

The **Direction Snap Modes** restrict the directions of the shift during stretch:

None: No restriction

Horizontal: The Y-shift value at the Stretch bar will be zero.

Vertical: The X-shift value at the Stretch bar will be zero.

Orthogonal: Stretch shift may be either vertical or horizontal. Either X-shift or Y-shift (or both) at the Stretch bar will be zero.

Diagonal: Stretch shift may be at any angle multiple to 45°. Either X-shift or Y-shift (or both) at the Stretch bar will be zero, or X-shift will be equal to ±Y-shift.

Hint: 1. Different objects may move differently during stretch, because they have different restrictions for movement. For example, if **Edit→Angle Mode→Preserve Angles Mode** is set, during stretch the movable parts of regions would be confined by their adjacent angles to unmovable parts, while the completely selected objects would move in the same direction as the mouse.

Hint: 2. If, during a stretch operation, some polygons change their topological structure (self-intersections, collinear segments, etc.), in some cases their elements are deselected from the stretch.

Hint: 3. If objects are picked for a stretch and some layers are hidden, the hidden objects would not be stretched. However, all objects remain picked, and if their layers are made visible again, they would be subject to stretching.

8.8: Jog Points

Jog Points are points on region/wire segments which divide that segment on separate parts. When objects are modified, the parts can be moved independently.

When you start the **Edit→Add Jog Points** operation, you activate a tool which allows you to draw a cut line on the layout. The tool behaves in the same way as the **Create Wire** tool: left-clicks add points, right-click finishes line creation. The line will "break" segments of regions and wires in points where the line intersects them. These points are called **Jog Points**. When you have added some Jog Points, use the Modify and Stretch tools to select parts of segments divided by Jog Points individually.

If you no longer need Jog Points you can select the **Edit→Remove All Jog Points** menu command. EXPERT also automatically deletes **Jog Points** when you no longer use the **Modify** or **Stretch** tools.

A common use of this operation is to split parallel buses of wires using 45° split lines. In that way, it is possible to simultaneously stretch all wires in the bus as shown in Figures 8-8 and 8-9.

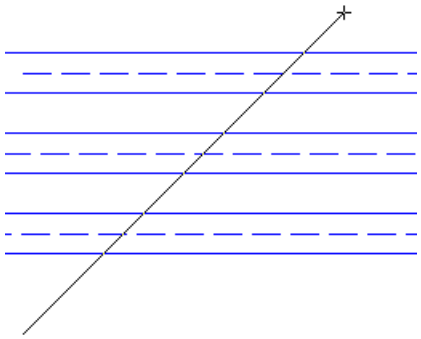


Figure 8-8: Example 1: Wire splitting

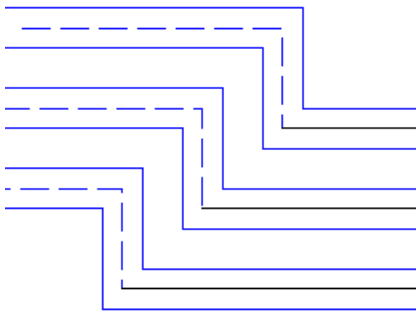


Figure 8-9: Example 1: Stretching of split wires

It is possible to add edges to a polygon, as shown in Figure 8-10.

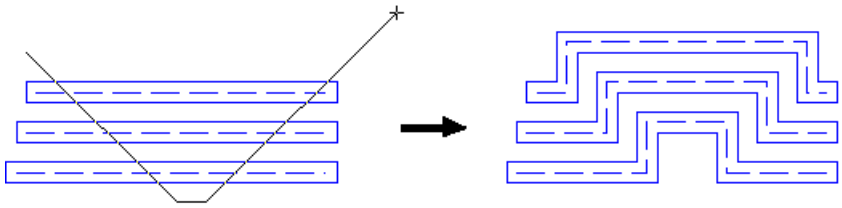


Figure 8-10: Example 2: Stretching of split wires

8.9: Inspecting/Changing Object Properties

When you select objects in layout view, the **Property Bar** shows common properties of all selected objects (of course you can select just one object and view properties of that particular object). You can change any attribute and user-defined property of objects and apply them by pressing the **Ok** button in the Property Bar.

You can navigate in the Property Bar either by using mouse (clicking on property you wish to change) or using arrow keys. Some properties (for example **Position**) have sub-properties. Such properties have a **"+"** sign near their names. To open the list of sub-properties, either click on "+" sign, or press right arrow key when the cursor is on such a property. To close the list of sub-properties, either click "-" sign or press left arrow when the cursor is on the top of the list. Use the up and down arrow keys to move the cursor between properties.

There are several classes of properties:

- **Labels** which you cannot edit.
- Properties which allow you to type in some **text**. Some of them accept only numeric values (for example X and Y Coordinates).
- Properties which allow you to **select from list** of values (for example Layer). When you click on such a property, a list of values appears. You can also open the list of values by pressing the **Enter** key when the cursor is on such a property.
- Properties which have a predefined list of values also allow you to type in your own values (for example Rotation).

Properties are edited in **dialog** boxes (for example Vertices). In such properties, the button with label "..." appears when they have focus. If you press that button then the appropriate dialog will be opened.

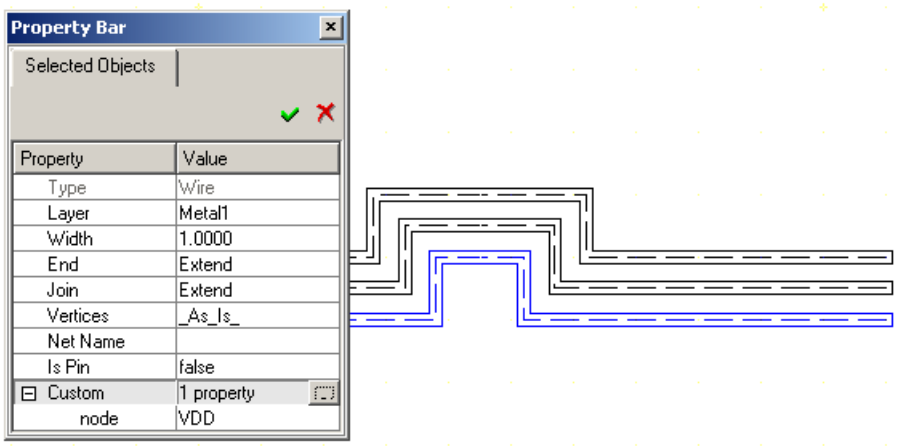


Figure 8-11: Access to object properties and attributes

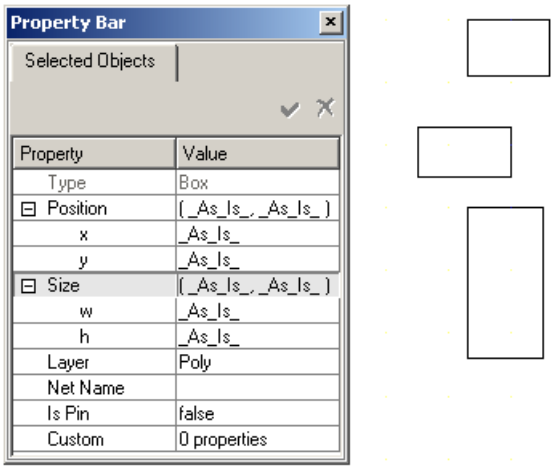


Figure 8-12: "As is" attribute values for selected boxes

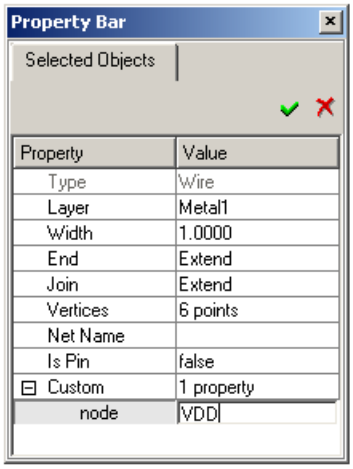


Figure 8-13: Access to user-defined properties

8.10: Searching Layout Objects

The menu command, **Edit→Search**, invokes a panel used to search objects in the layout according to various criteria. This panel is shown in Figure 8-14.

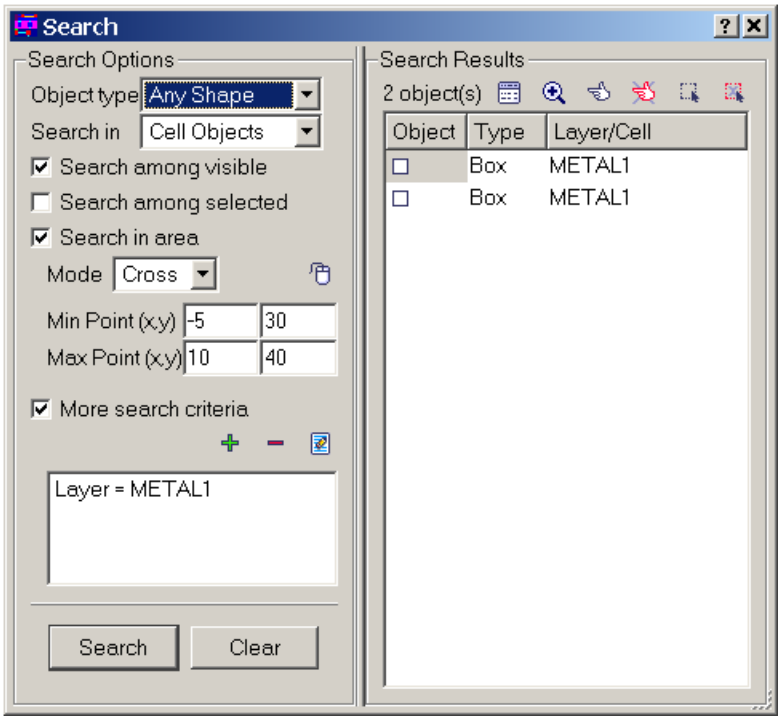


Figure 8-14: Search Operation

The **Search** dialog has two panels named **Search Options** and **Search Results** correspondingly. You can change the size of each panel or hide any of them by clicking and dragging the splitter. **Search Options** panel is used to specify search criteria.

It is possible to search objects according to various criteria. Some basic criteria are shown directly on the **Search Options** panel.

Object type: In addition to primitive object types (box, wire, region, ellipse, text), the following types may be specified, which deserve some explanation:

- **Any Shape**: to search among primitive geometric objects: boxes, wires, ellipses, regions.
- **Instance**: to search among cell instances and instance arrays.
- **Array**: to search among cell instance arrays only.
- **Any Object**: to search among all objects mentioned above.

Search in modes: A search may be performed among the current **Cell Objects**, as well as in the **Cell Hierarchy** of the current cell.

Search among selected and **Search among visible** options are self-explanatory.

The option **Search in area** allows you to search for the objects in the specified area. If this option is checked, the **Search in area** group appears. The area to be searched may be specified with the mouse by clicking the **Define by mouse** button and drawing a selection box, or manually by typing the values in the **Min Point** and **Max Point** input fields. In addition, there are two modes to define how the desired objects must be related to the specified search area:

- **Inside:** to search among objects completely inside the search area.

- **Cross:** to search among objects intersecting the search area.

The option **More search criteria** is used to specify additional search criteria in terms of object attributes and user-defined properties. If this option is checked **More search criteria** group is added to the **Search Options** panel. Use the buttons **Add new criterion**, **Edit selected criterion** and **Delete selected criterion** to specify and modify additional search criteria. You can add any number of such criteria; each of them narrows down the search. Many search criteria from this group are different for different types of objects. A typical set of such criteria is shown in Figure 8-15 for wires.

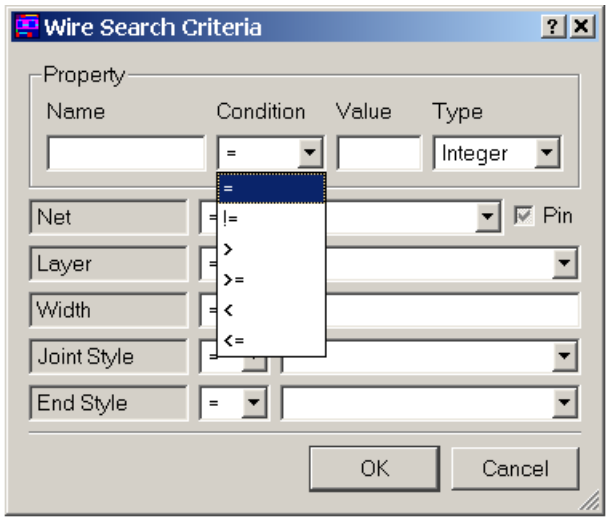


Figure 8-15: Search criteria for wire

As a rule each criterion consist of three parts: **Name**, **Applicable condition** and **Value**. In addition, for user-defined property criterion it is necessary to select **Type**, for proper processing of the condition (comparison or substring). For some criterion, for example **Pin**, it is needed to check or uncheck the corresponding option.

With the exception of user-defined properties, the **Name** part is predefined ("Layer", "width", etc.) The **Condition** is selectable from the drop-down list, and its meaning is usually clear, with one explanation. The "**filter**" condition may be applied to textual attributes, such as layer and cell names, text strings and string values of user-defined properties. In the current version of EXPERT a simple version of filter is implemented, the same as in cell selectors: the star (*) character replaces a substring of any length and the question mark replaces a single character. For example, the condition:

```
"Cell name =(filter) *_v.01"
```

will search for instances of all cells with names ending with "_v.01". The condition:

```
"Layer =(filter) Met?"
```

will search for objects in layers Met1, Met2, etc., but not in Metal.

Note: 1. You may add a criterion with Property Name given without Property Value. In this case all objects with any value of the given property will be found.

Note: 2. You may use wildcard characters in the names of user-defined properties, as well as in their values when doing search by Edit→Search command. For example, by typing "*" both as a name and "*" as a value while selecting "filter" as search condition type, you may find all objects with at least one user-defined property.

When all required search criteria are set, click the **Search** button to perform a search. When the search is finished, the **Search Results** panel shows the number and list of found objects. If the number of object exceeds 500 then the first line of the object list will be "**Too many items. First 500 of them were listed here. Double click here to show all items**." But be aware that a large amount of lines in the result list may slow down its performance.

As you click on any line in the list of object the **Property Bar** will show properties of the object listed in the highlighted line. You can modify any object attribute by editing it in the **Property Bar** (see Section 8.9: "Inspecting/Changing Object Properties"). If you select more than one found object, the common properties of all of them will be shown. Use the button **Highlight All** to inspect and edit the common properties of all found objects.

The buttons **Select highlighted objects** and **Deselect highlighted objects** allows you to select/deselect objects corresponding to highlighted lines in the list of found objects shown in the **Search Results** panel. The buttons **Select all objects** and **Deselect all objects** correspondingly select and deselect all found objects.

Note: A single run of "Search" finds objects such that each of them satisfies ALL search criteria. In other words, a single search answers queries such as: `<Find all wires in layer Metal2 with property "Node" equal to "VDD">.`

If, for example, you want to find all wires **and** all boxes according to certain criteria, you must do the following:

- find wires
- click "Select all" at the "Find" panel
- find boxes
- click "Select all" at the "Find" panel

Now you have selected all wires and boxes that satisfy your criteria.

8.11: Other Editing Commands

The Edit menu command also contains some setup features, which allows you to toggle on and off certain functionality in the current session.

Edit→Gravity allows you to activate or deactivate gravity. More details about the Gravity setup can be found in the **Setup→Editor/Viewer...→Gravity** form and in Chapter 11, section 11.3.

Edit→Snap Cursor to Grid allows you to specify the snapping mode of the cursor in relation to the grid. Section 3.1.3 of Chapter 3 and Chapter 11 "Setup" provides a more completed description of this feature.

The submenu **Edit→Region Mode** provides you with four different settings for geometry creation; Normal, Merge, Hole and Hole Selected.

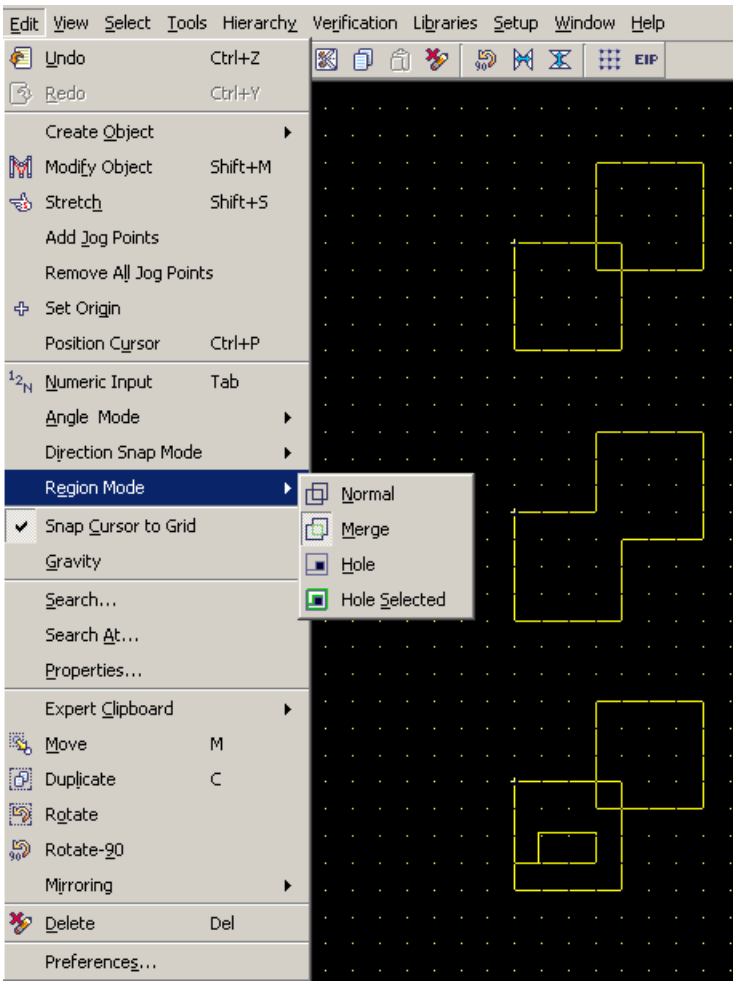


Figure 8-16: Region Mode

In Normal mode each geometries is drawn independently from each other. This mode is shown in the top right of Figure 8-16 where two rectangles overlapped each other and remain two different entities.

In Merge mode the same two overlapped rectangles merge to create a single geometry. This single entity can be seen in the center right of Figure 8-16.

The Hole mode is used when you want to create hole in geometry. This mode is illustrated in the bottom of Figure 8-16, where the lower rectangle was modified using the Hole mode.

The last Region mode available to you is Hole Selected. This mode is similar to the Hole mode to the exception that it will act only on selected geometries.

Edit→Preferences opens the Preferences window allowing you to set various options and parameters in EXPERT. This same window can also be access by the **Setup→Setup...** command.

8.12: Edit Submenu Operations

8.12.1: Stitch Wires

The wire stitching tool allows you to easily change a layer of particular segments of wires. It automatically creates appropriate contacts, connecting these segments with other segments on different layers. To activate this tool select the **Edit→Stitch Wires** menu option.

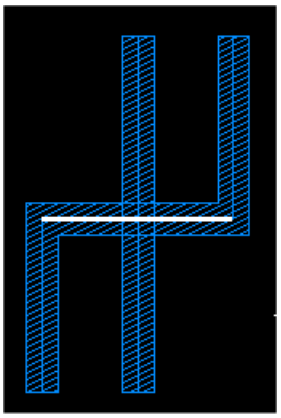


Figure 8-17: Selected Wire Segment

You can select wire segments by clicking on them, or dragging the selection frame just like in the Stretch tool. Then you can change the layer of selected segments by selecting a different layer in the **Layer** field in the **Numeric Input** panel, or in the context menu which can be activated by clicking the right mouse button.

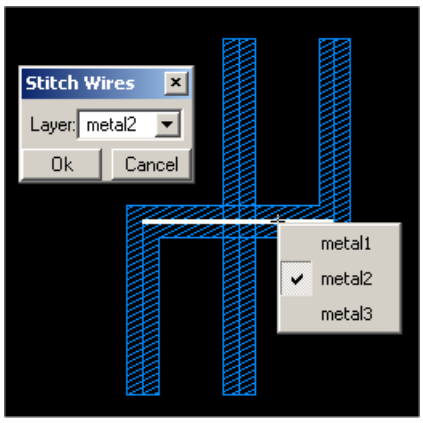


Figure 8-18: Changing the Layer of a Selected Segment

Alternatively, you can double-click on selected wire segments, and the layers will be changed in cycle.

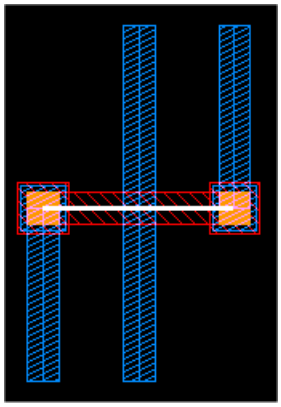


Figure 8-19: Selected Segment Moved to Different Layer

Note: You need appropriate contacts to be defined in technology in order to use this tool.

8.12.2: Align

There are two groups of commands in the **Edit** menu to help align sets of objects: **Edit→Align** and **Edit→Space Evenly**. These commands affect the selected objects only. A description of commands from the **Align** sub-menu is in Table 1, and from **Space Evenly** in the Table 2.

Table 1: Align Commands

Align	Description
Lefts	Move selected object to the left until their left edges are on the same line
Centers	Move objects in horizontal directions until their centers has same X coordinate
Rights	Move objects to the right until their right edges are aligned to the same line
Tops	Move objects up until their upper edges has same Y coordinate
Middles	Move selected objects in vertical directions until their centers a on the same line
Bottoms	Move objects down until their bottom edges are aligned to the same line

Table 2: Space Evenly Commands

Distribute	Description
Across	Move selected in horizontal directions until gaps between each two neighbor objects are the same
Down	Move objects in vertical directions until distances between any two consecutive objects are the same

[This page is intentionally left blank.]

Chapter 9:
Viewing the Layout

9.1: Layout Windows

EXPERT provides a multiwindow working environment. You can see and edit various parts of the layout through their own windows. This chapter describes how to open, arrange, and close layout windows, as well as how to navigate inside a layout window.

9.1.1: Opening/Closing Windows

A layout window in EXPERT is opened to view a cell. You may open several cells in their own windows, as well as several windows for the same cell.

Below is a summary of commands that may open/close layout windows. Detailed descriptions may be found in the corresponding sections of the manual.

The **New...** command of the **File** submenu closes the current project (if any) after the name for the new project being created has been specified. If the current project was modified, you will be asked to save the changes.

The **Open...** command of the **File** submenu closes the current project (if any) after you specify the name of the previously created project to open. If the current project was modified, you will be asked to save the changes. The project to open may be of EXPERT'S `.eld` format, `.sld` format, as well as from **GDSII**, **Applicon**, **CIF**, **LEF** or **DEF** format.

The **New...** command of the **Cell** submenu creates a new empty cell.

The **Open...** command of the **Cell** submenu opens an existing cell.

Some operations that may create cells have the **Open now** option, so that a created cell is immediately opened.

The list of opened layout windows, identified by their cell names, can be found in the **Window** submenu, or by **View→Dock Windows→Windows**. Only one of these windows may be active, and you can navigate across the layout shown in it and edit this layout. Some of the windows may be partially visible but inactive, and some invisible.

You must activate the window before working within it. You can either pick the name of the required window in the list of the **Window** submenu or **Windows** dock windows, or pick this window directly on the screen by mouse if it is partially visible, or use the **Ctrl-F6** combination of keys to rotate through the window list.

You can close the current active window by selecting the **Close** operation of the **Window** submenu, or by pressing the close icon in the **Windows** dock windows. A visible window may be closed by clicking the **X** button in the top right corner of the window.

All opened windows may be closed by using the **Close All** operation of the **Window** submenu, or by pressing the close all windows icon from the **Windows** dock windows.

The **underline** button, located at the top right corner of a window, permits you to minimize the window into an icon. The icon is placed at the bottom of the EXPERT'S working area, but may later be dragged anywhere in this area.

More advanced window managing features are provided through the **Cell Map Window** environment, described in Section 9.6: “Cell Map” of this chapter.

9.1.2: Window Arrangements

To simplify the management of opened windows, the **Window** submenu and the **Windows** dock windows provide features that automatically arrange windows to suit your needs.

The **Cascade** operation in the Window submenu arranges windows in overlapping order. Only the nearest window is visible - only titles of other windows can be seen, (see Figure 9-1). The cascade operation can also be performed by pressing the corresponding icon in the **Windows** dock windows.

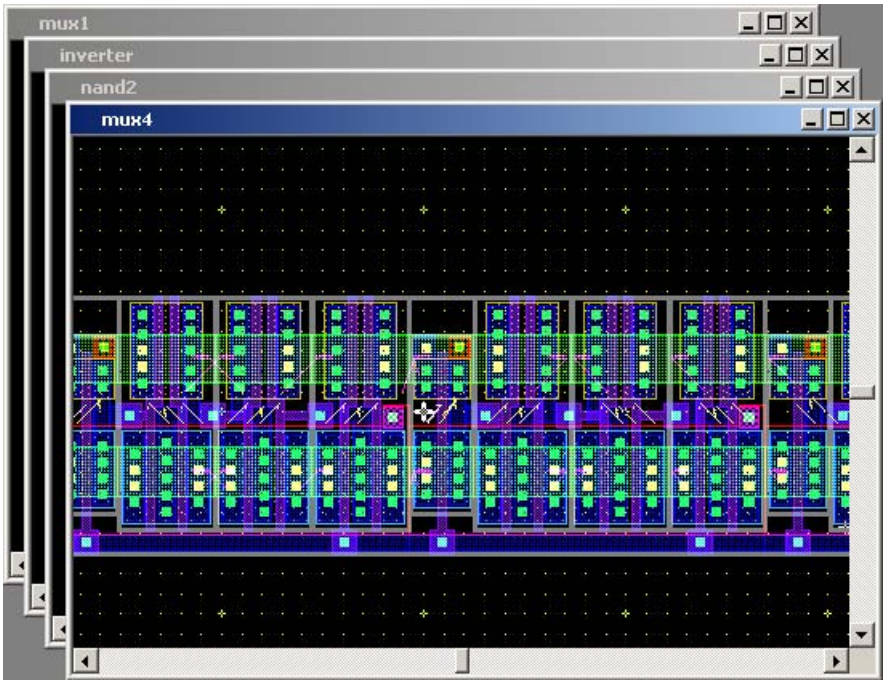


Figure 9-1: Cascade Method

The **Tile**, **Tile Vertically** and **Tile Horizontally** operations arrange windows to make them all fully visible, (see Figure 9-2, Figure 9-3 and Figure 9-4). Pressing the corresponding icons in the Windows dock windows will also launch the desired Tile command.

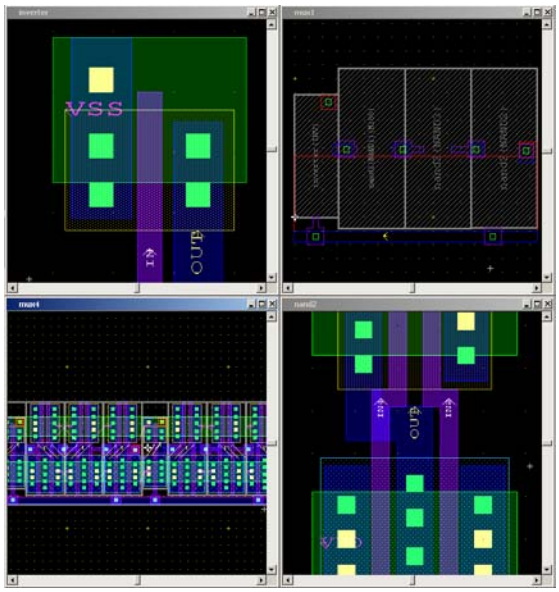


Figure 9-2: Tile Method

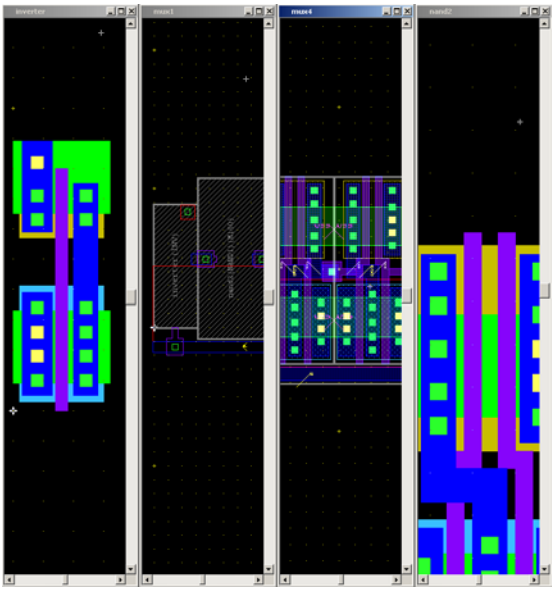


Figure 9-3: Tile Vertically Method

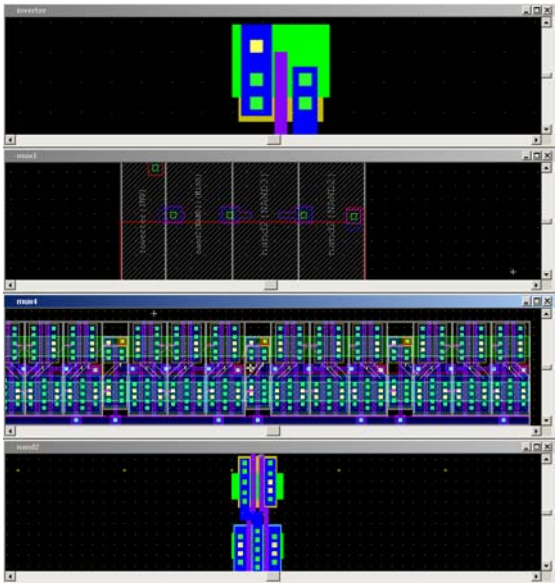


Figure 9-4: Tile Horizontally Method

9.2: Filling Modes

Shape layout objects (region, box, circle/ellipse, wire) may be drawn in three filling modes: **solid, stipple** and **wireframe,** see Figure 9-5. The default filling mode of drawing may be selected either from the **View→Filling** submenu, or by buttons at the **Filling toolbar**.

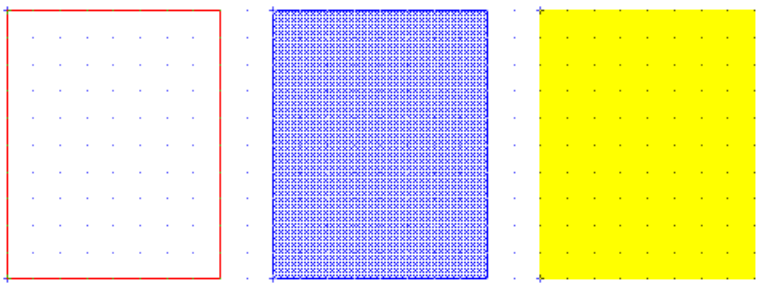


Figure 9-5: Filling Modes: Wireframe, Stipple, Solid

Stipple patterns are assigned to layers in the layer setup (**Setup→Technology→Layer/Wire Setup...** menu command). There are two sets of predefined stipple patterns (System, X-Stipples), as well as a stipple editor that allows you to create your own stipple patterns and store them in a file (see Chapter 11: "Setup", Section "Stipple Setup").

See also Chapter 15: "Plotting" which provides information about filling modes on the plot.

9.2.1: Individual Filling Modes for Layers

It is possible to set filling modes for individual layers that override the common default filling mode. A simple way to do this is to right click at the color strip of the layer in the **Layer bar** dock window. You will see a list, as shown in Figure 9-6. You may select either one of three modes or the “default”. In the latter case, the common default setting will be applied to the pointed layer.

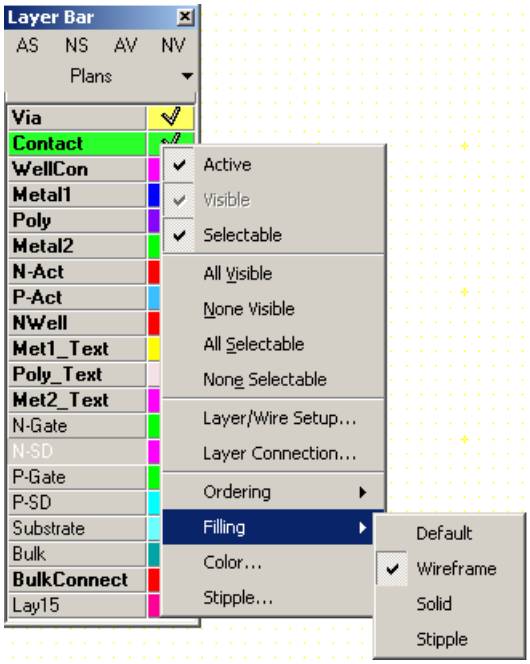


Figure 9-6: Individual filling mode selection

A more advanced way to set/change individual filling modes is described in Chapter 11: “Setup”, Section “Layer Plans: Filling, Selectivity, Visibility, and Inclusion”.

9.2.2: Individual Filling Modes for the Active Layer

The **View→Filling→Active Layer Mode** submenu is used to assign the filling mode for the active layer. One of the four values may be selected: wireframe, solid, stipple, and no change. This setting defines the filling mode of the active layer. When the layer is activated, its filling is controlled by this setting. When the layer is deactivated, its filling mode is restored to the previous one. For example, it is sometimes convenient to see all layers in wireframe, and the active layer is always highlighted by being drawn filled. This setting is repeated in the **Setup→Editor/Viewer...→Viewing** form.

9.3: Panning Commands

Purpose

Shifts the image inside the current window. It permits you to make visible the required neighborhood of the edited area, or to roam across the layout.

There are several ways of panning in the **View→Pan** submenu (see Figure 9-7).

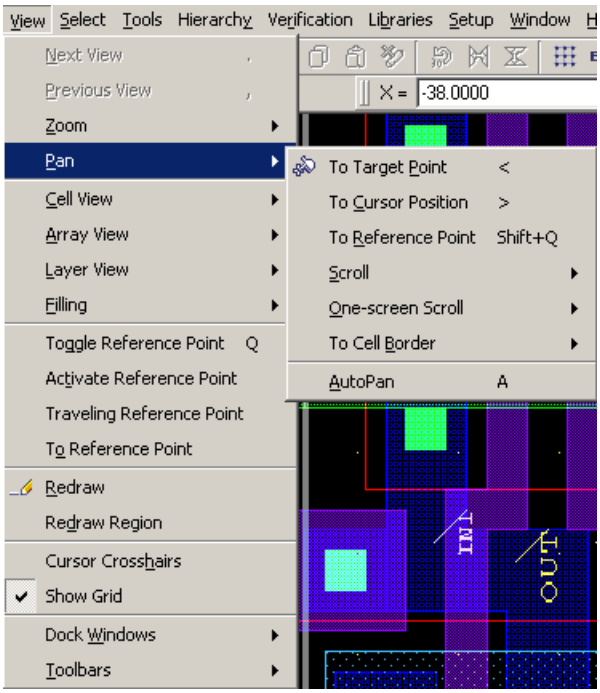


Figure 9-7: View Submenu

9.3.1: Pan To Target Point

The **Pan→To Target Point** operation sets a point of the layout to be the new center in the active window.

Upon this command, the cursor assumes the “Pan” shape and you must click the mouse at the required new center of the window. Figure 9-8 shows the visible part of the layout before and after completing the **View→Pan→To Target Point** operation.

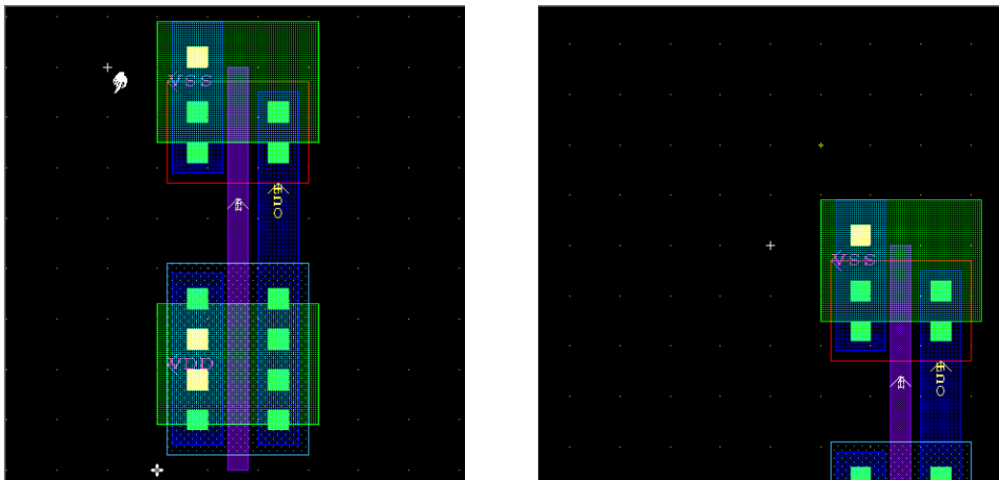


Figure 9-8: Pan→To Target Point operation

9.3.2: Pan To Cursor Position

The **Pan→To Cursor Position** operation allows you to pan to the current position of the cursor. An easy way to use this command is by means of the keyboard shortcut. Once a keyboard shortcut in **Setup→Editor/Viewer...→Shortcut** is specified, placing the cursor at the desired position and hitting the shortcut key will pan the screen to the cursor.

9.3.3: Pan Scroll Left/Right/Up/Down

The **Pan→Scroll→Left/Right/Up/Down** operations shift the boundaries of the visible area of the layout in the corresponding direction. The size of the shift is defined by the **Scrolling steps in pixels** value of the **Setup→Editor/Viewer...→Viewing** form, (see Chapter 11: “Setup”, Section 11.3: “Editor/Viewer Form”). The maximum value considered is 50.

You may also use the vertical and horizontal **scroll bars** to move the visible window over the layout. Finally, you may use the **Cell Map Window** described in Section 9.6: “Cell Map”.

9.3.4: Pan To Reference Point

The **Pan→To Reference Point** command pans to the temporary reference point, if it is set (even if it is inactive at the moment).

9.3.5: Pan Autopan

The **Pan→Autopan** menu command acts as a toggle, and turns on/off the autopan mode. If this mode is on, and EXPERT is in any drawing mode (object creation, ruler, zoom box, etc.), then when the cursor comes close to the border of the layout window, the layout scrolls automatically. The “autoscroll threshold” (how close the cursor must be to the border) is 50 pixels. The scroll step determines the distance the autoscroll will move the window, once the threshold is reached.

9.4: Zoom

Zoom features allows you to increase or reduce the visible sizes of the cell layout in the window so that you may either investigate small details of the layout structure, or overview its parts in general. Several ways of zooming are provided by the **View→Zoom** submenu: **Zoom Into Window**, **Zoom Out of Window**, **Zoom In**, **Zoom Out**, **Zoom To Fit**, and **Zoom Selected**.

In addition, an easy control over the visible area is provided by the **Cell Map Window**.

9.4.1: Zoom to Fit

Purpose

Shows the whole layout of the active cell. Figure 9-9 demonstrates the contents of the active window before and after the **Zoom To Fit** operation.

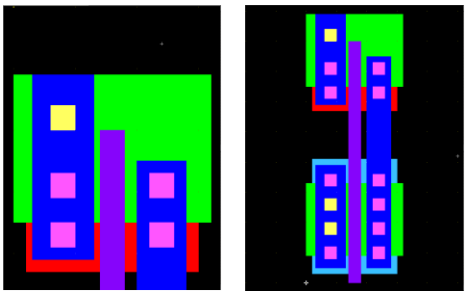


Figure 9-9: Zoom to Fit

9.4.2: Zoom Selected

Purpose

The menu command, **View→Zoom Selected**, zooms from the current window to the selected objects in the edited cell. (Remember that during EIP, the edited cell is different from the current cell).

9.4.3: Zoom Into Window

Purpose

Replaces the image of the current window with the part of the layout located inside the selected box area. There are two ways of zooming to a window.

First, you may click at any two opposite corners of the required **zoom box**.

Alternatively, it is possible to define the zoom box by "drag" mouse action. Depress the left mouse button at one point, drag the mouse to the second point, then release the button.

The center of the new image window is located at the center of the chosen box, and its resolution is magnified so that the drawn zoom box covers the whole window.

Figure 9-10 demonstrates the **Zoom Into Window** operation.

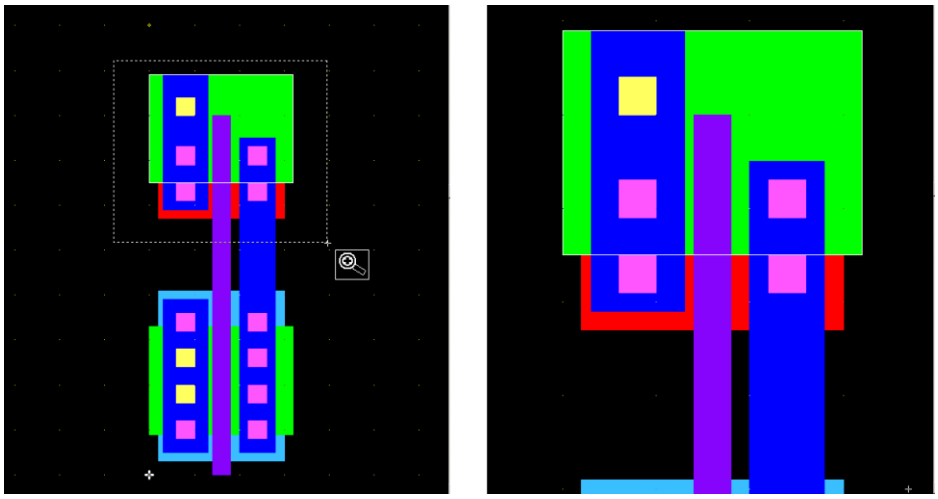


Figure 9-10: Zoom In to Window

9.4.4: Zoom Out of Window

Purpose

Replaces the image of the current window with a zoomed out version.

The box drawn with the use of the **Zoom Out of Window** tool, indicates the size that the current layout view will occupy in the next view of the window. There are two ways of zooming out of a window.

The user may click at any two opposite corners of the required **zoom out box**.

Alternatively, it is possible to define the **zoom out box** by “drag” mouse action. Depress the left mouse button at one point, drag the mouse to the second point, and then release the button.

The current view will be relocated where the **zoom out box** was drawn.

Figure 9-11 shows the layout before and after the **Zoom Out of Window** operation.

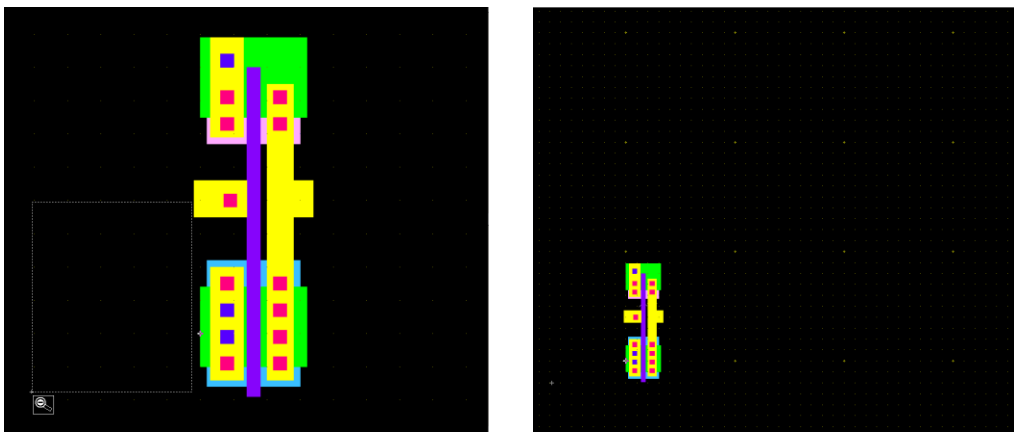


Figure 9-11: Zoom Out of Window

9.4.5: Fixed-Ratio Zoom In and Out

Purpose

The **Zoom In** and **Zoom Out** operations increase and decrease the resolution of the image inside the current window, respectively. The center point of the visible part of the image remains in place. These operations permit you to accurately investigate or edit layout details located at the window center (**Zoom In**), or to see a wider neighborhood of the currently visible part of layout.

The resolution changing rate is pre-defined, which makes these operations easy to use. From the other side it can be modified from the **Setup→Editor/Viewer...→Viewing** form (**Zoom Factor** value). See Chapter 11: "Setup", Section 11.3: "Editor/Viewer Form" for more information.

Figure 9-12 shows the layout before and after the **Zoom In** operation (**Zoom Factor** = 2).

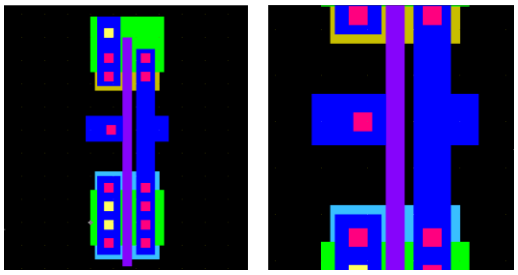


Figure 9-12: 2x zoom in

There are **plus** and **minus** buttons at the Command bar for zoom in/out. The default shortcuts are "Gray plus" and "Gray minus" keys at the numeric keypad of the keyboard.

9.5: Mouse Wheel for Scrolling and Zooming

- If no control keys depressed, the mouse wheel scrolls the layout up and down.
- If the **Alt** key is depressed, the wheel scrolls left/right.
- If the **Ctrl** key is depressed, the wheel zooms in/out.
- If the **Shift** key is additionally depressed, the scroll factor is made 15 times more.

9.6: Cell Map

Cell Map window can be activated by selecting menu command **View→Dock Windows→Cell Map**.

This window performs following two functions:

- it shows the preview of current cell;
- it displays the frames which indicate the scrolling area of every opened window of this cell.

If there are several opened windows of current cell, user can activate any of them by clicking on corresponding scrolling frame. It is also possible to drag the scrolling frames by mouse and the referred window will be scrolled respectively.

Usually, **Cell Map** window displays the currently opened cell but it can also display other cells, for instance, currently selected cell in **Projects Tree** or in any other tree or list of cells.

As any other docking window, **Cell Map** window can be docked and undocked to available docking areas for user convenience. The user can also resize it as required to satisfy his/her current needs.

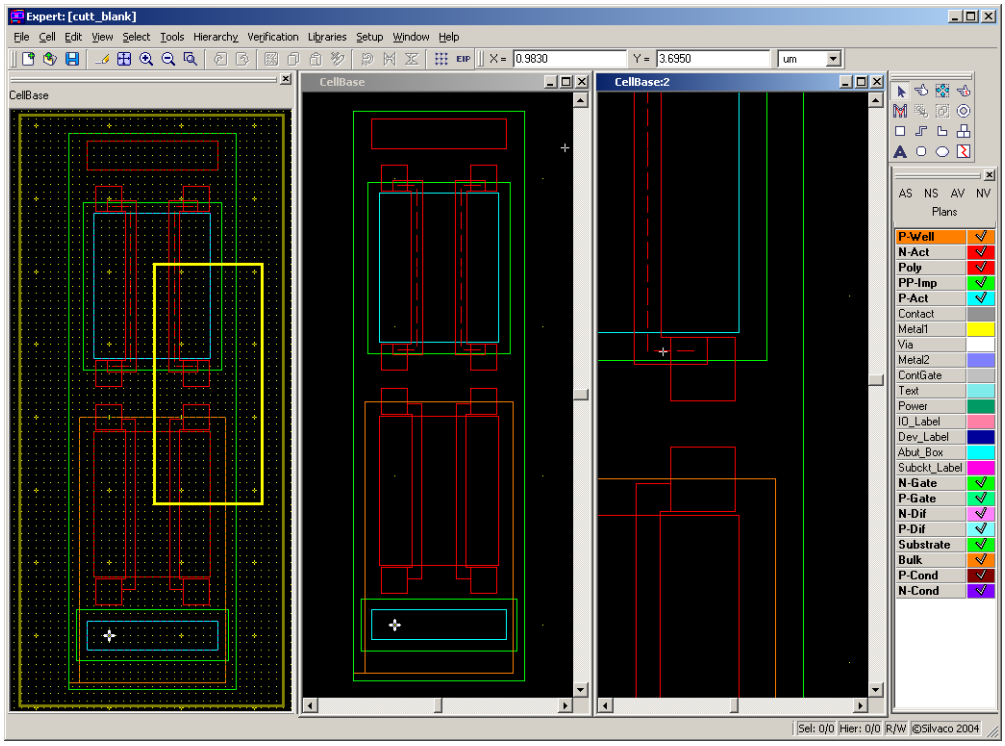


Figure 9-13: Cell Map Window

9.7: View Cell View

The **View→Cell View** submenu controls the degree of detail in the **edited cell** layout.

Settings assigned from this submenu are applied to the **edited cell** in the **active window**. At the same time they become actual for this cell in all windows where it is opened, either normally or by entering from **Edit-in-Place**.

There are other ways to change the contents of layout's view. It is possible to make parts of the geometry **hidden** for viewing convenience. This is controlled by the following settings:

- Layer visibility settings in **View→Layer View** submenu or in the **Layer Bar**.
- **View→Array View** menu command,
- **Setup→Editor/Viewer...→Viewing→**"Draw layout for at most <N> hierarchy levels".

9.7.1: Flat

In the **View→Cell View→Flat** mode, only the real geometry of the layout is shown; no cell instances/arrays are denoted in any way. In other words, the **edited cell** looks like with its hierarchy is flattened.

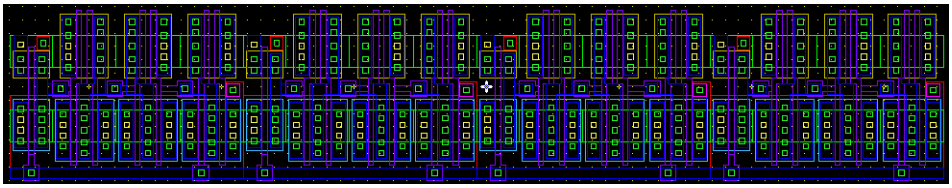


Figure 9-14: Flat View

9.7.2: Lazy

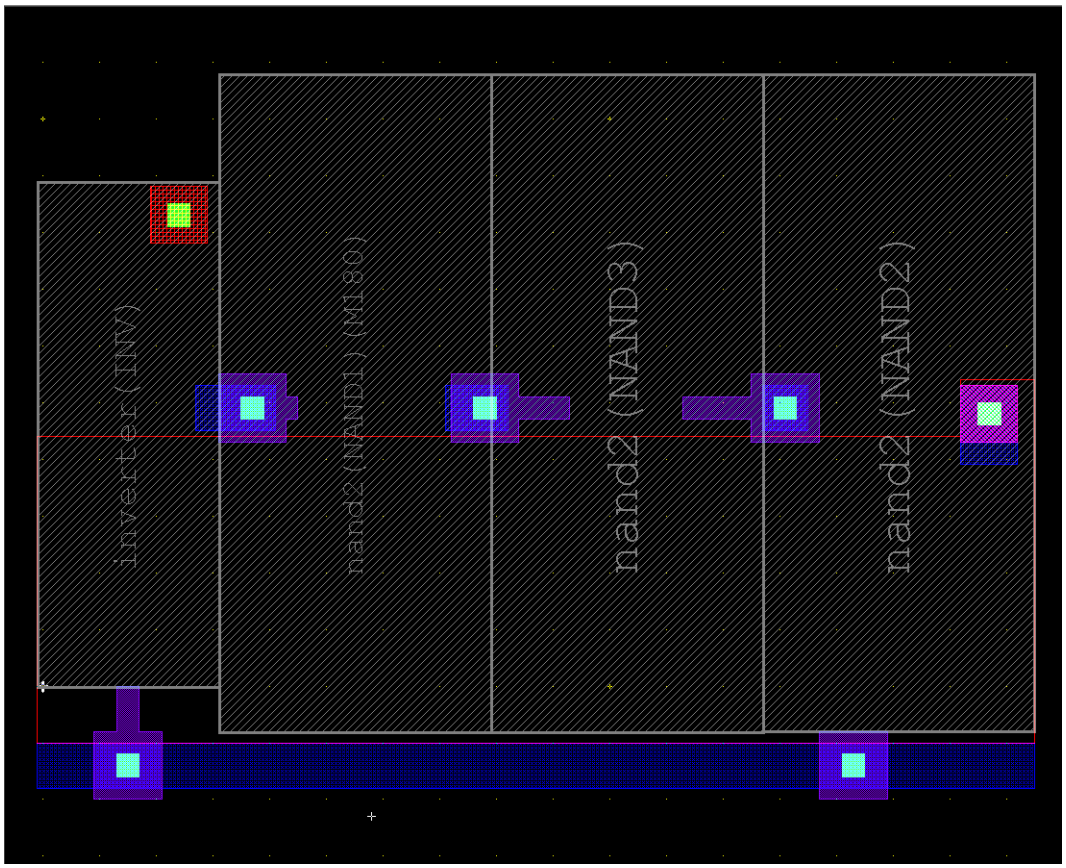


Figure 9-15: Lazy View with Edited Cell Primitives

In the **View→Cell View→Lazy** mode, only primitives and instances/arrays immediately belonging to the **edited cell** are shown. Instances/arrays are displayed as hatched boxes (i.e., with their insides hidden). In other words, only the top level of the **edited cell** hierarchy is displayed, the rest of cell hierarchy is hidden.

When cell instance insides are not shown (Lazy View or other views with restricted level of display), the cell name is written within the instance's frame.

Lazy view shows only the layout objects which are directly selectable/editable in the edited cell.

9.7.3: Floorplan Mode

In the **View→Cell View→Floorplan Mode** mode only instances/arrays are displayed, by their bounding boxes, depending on hierarchical depth selected in the **View→Cell View→Hierarchy Depth** submenu.

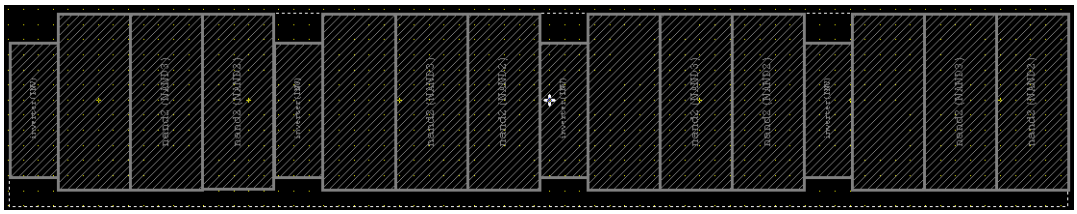


Figure 9-16: Floorplan Mode

9.7.4: Primitives Mode

In the **View→Cell View→Primitives Mode** mode only primitives are displayed.

9.7.5: Edited Cell Frame

If the **View→Cell View→Edired Cell Frame** menu option is checked, the bounding box of the edited cell is shown as a dashed frame around the cell, or around the instance if the edited cell is the cell under edit-in-place, otherwise it is hidden. You may toggle this option on and off by clicking the left mouse button in this menu item. Figure 9-16 illustrates the bounding box when the option is on.

9.7.6: Hierarchy Depth of Display

It is possible to restrict the number of hierarchy levels shown on the screen.

The **View→Cell View→Hierarchy Depth** submenu allows you to specify the deepest of hierarchy levels whose primitives are displayed (Lazy view corresponds to zero depth). There are seventeen possible values for depth. However, you can go beyond them by using the **Increase/Decrease** depth commands.

The **View→Cell View→Increase Depth** and **View→Cell View→Decrease Depth** commands are changed by the display level set by the command **Depth** or **Lazy**. These commands are most convenient when executed by shortcuts. Clicking the shortcut several times changes depth by several levels.

When cell instance inside are not shown (Lazy View, or other views of limited hierarchy display depth), the cell name is written within the instance's frame.

Option **Setup→Editor/Viewer...→Viewing→Draw layout for at most <N> hierarchy levels** sets the number of hierarchy levels drawn, starting from the deepest level set by the above commands.

9.7.7: Activate Layout View

The **View→Cell View→Active Layout View** command allows you to activate the layout window while working in the numeric input mode. This command is best used if assigned to a shortcut.

Note: Another important view of cells is the Xi Script of parameterized cells. The Xi Script of a Pcell currently active in the layout window can be display using the command **Cell→Pcell Script**.

9.8: View Array View

The **View→Array View** menu item allows you to select one of the three available styles of array display (see Figure 9-17). These array styles are effective only with **Flat** view style and apply to all arrays in all windows.

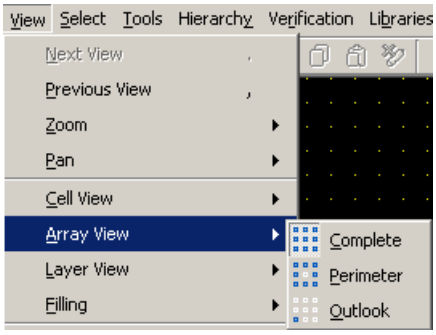


Figure 9-17: Array View Modes

The **View→Cell View→Edited Cell Frame** option adds the gray frame around the array shown in the Outlook mode. The styles Complete, Perimeter and Outlook are also show in Figure 9-18.

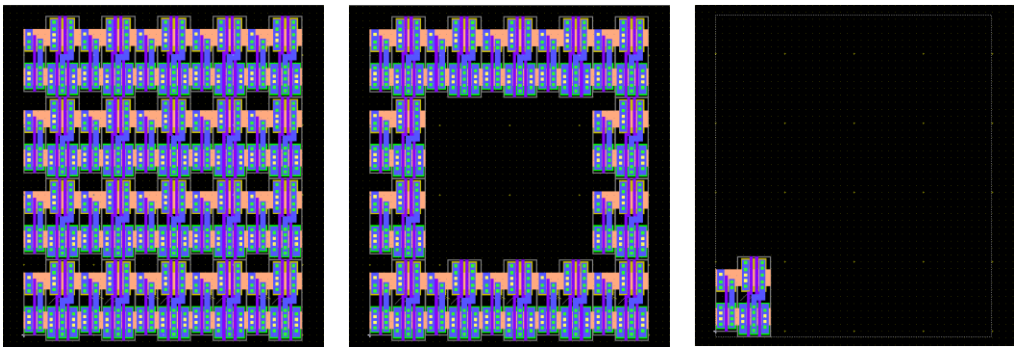


Figure 9-18: Styles of array display

9.9: Layer View

Commands from the **View→Layer View** submenu, with the exception of the ones described below, are described in Chapter 11: "Setup", Section 11.4.3: "Layer/Wire Setup".

9.9.1: Ordering

Layer View→Ordering

- Technology Order
- Alphabetical
- GDSII number
- Inverse order

These options are duplicated in the layer bar context menu.

9.9.2: Layer Plan

The **Layer View→layer Plan** menu is used to invoke the **Layer Plan Setup** window or to **Fix the Layer Plan**.

This Setup window allows you to set the following properties for each layer:

- Individual filling settings.
- Visible/Invisible in the layout.
- Selectable/nonselectable (i.e., elements of this layer may be selectable or not) Note that invisible layers are non-selectable as well.
- Included/not included into the layer list of the **Layer** bar.

This command is also accessible by pressing the Plan button located at the top of the layer bar.

More on **Layer Plan Setup** can be found in Chapter 11: “Setup”.

9.10: Temporary Reference Point

With this feature it is possible to see the cursor position at the **Metric** bar with respect to a selectable reference point. In addition, if a temporary reference point is active, the distance of the current cursor from the reference point is displayed in polar form. Distance and Angle are in the bottom right of the EXPERT'S window.

9.10.1: Toggle Reference Point

The **View→Toggle Reference Point** menu command allows you to set/remove (toggle) a reference point. It is useful only when a shortcut is assigned to it. The default shortcut key is “Q” (it may be reassigned).

If there was a reference point active, hitting “Q” deactivates it, and the text at the Metric bar reads: **X= ... Y = ...**.

If there was no reference point, hitting “Q” makes the reference point at the current cursor position. The text at the Metric bar reads: **RX= ... RY = ...**, where **R** is for Relative.

Hint: If you want to move a reference point, hit the “Q” key twice with mouse cursor at the new position.

9.10.2: Activate Reference Point

The **View→Activate Reference Point** menu command allows you to switch between the set temporary reference point and the origin as position reference.

- If the **Activate Reference Point** menu option is not checked, the origin is the reference point, and the text at the Metric bar reads: **X= ... Y = ...**.
- If the **Activate Reference Point** menu option is checked, the cursor position is shown relative to the new reference point. The text at the Metric bar reads: **RX= ... RY = ...**, where **R** is for Relative.

9.10.3: Traveling Reference Point

If the **View→Traveling Reference Point** mode is on and the temporary reference point is activated, then during creation of boxes, wires and regions, the reference point is always moved to the last clicked point.

9.10.4: To Reference Point

The **View→To Reference Point** command moves the cursor to the reference point.

9.10.5: Distance from Reference Point

When a temporary reference point is activated, the distance from the cursor position to the reference point is shown at the bottom right of the EXPERT'S window. Recall that the Angle of the current cursor from the reference point is also displayed.

9.11: Various

9.11.1: Redraw

The **View→Redraw** command refreshes the image in the current window.

9.11.2: Redraw Region

The **View→Redraw Region** command refreshes the image in the selected box.

9.11.3: Cursor Crosshairs

When the **View→Cursor Crosshairs** setting is on, the cursor is supplied with dashed crosshairs. They help to align different objects in horizontal and vertical directions.

9.11.4: Show Grid

The **View→Show Grid** menu command toggles on/off the visibility of the grid in the layout screen. This option is also accessible by **Setup→Editor/Viewer...→Grid Setup** option visible.

9.11.5: Dock Windows

The **View→Dock Windows** menu command allows you to specify which dock windows to display. Check/ uncheck the dock window in this list and see the immediate changes in the screen layout.

More on dock windows can be found in Chapter 3: "User Interface", Section 3.1: "Expert Interface" and in Chapter 11: "Setup".

9.11.6: Toolbars

The **View→Toolbars** menu command allows you to specify which toolbars to display. Check/ uncheck the toolbar in this list and see the immediate changes in the screen layout.

More on toolbars can be found in Chapter 3: "User Interface", Section 3.1: "Expert Interface" and in Chapter 11: "Setup".

Chapter 10:
Hierarchy Operations

10.1: Hierarchy Menu

This chapter describes how EXPERT commands allow you to operate with the chip's hierarchy. To operate these commands, use the Hierarchy submenu (Figure 10-1).

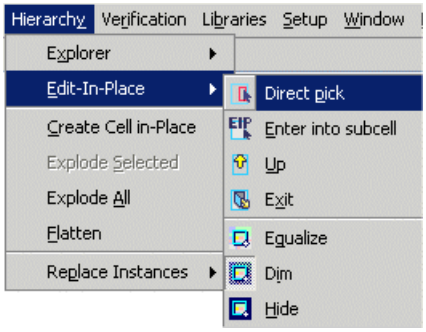


Figure 10-1: Hierarchy Submenu

They may be split into the following catagories:

- The **Explorer** commands provide the means to view and walk through the hierarchy tree, and to reach layout objects, regardless of their position in the chip's hierarchy.

† • The **Edit-in-Place** commands allows you to edit a cell while being within its particular instance at any level of hierarchy, and to view its surroundings in the chip.

Note: In EXPERTVIEWS, the Edit-in-Place group of commands is replaced by Show-in-Place commands.

---

† • The Hierarchy modification commands allow you to modify the chip's hierarchy of cells, either by preserving its actual geometry or by preserving the hierarchy tree structure.

The most important commands from hierarchy edition and edit-in-place groups use shortcuts via buttons from the toolbars: the **Hierarchy bar** and the **Edit-in-Place** bar.

10.2: Hierarchy Navigator (Project Tree)

Purpose

Invokes a submenu of Hierarchy Navigator (**Project Tree**) tools for navigation through the chip hierarchy and for display of cell instances within the hierarchy tree.

Note: If you are in the cell or layer list, you can type the first letter of the name (of the cell or layer), and the list will scroll to the next element of the list starting with this letter. The same is valid for an expanded **Project Tree**. Moreover, you may type in several first letters of the name with sufficiently short delay between the keystrokes. This narrows the selection further.

10.2.1: Project Tree

The **Project Tree** window can be activated by the menu command **Hierarchy→Explorer→Project Tree**.

Project Tree is a tool for navigating through the hierarchy of the chip design represented in the form of a tree. In addition to the hierarchy tree navigation, **Project Tree** provides a convenient, visual supplement for other layout editing and navigation operations that are somehow related to chip hierarchy.

There are two available options on the toolbar of the **Project Tree** window. The option **Show all projects** shows the hierarchy for all projects. The option **Show active cell** always displays the hierarchy of active cell.

Chip hierarchy can contain a huge number of elements, that's why the actual displayed number of elements in any expanded node is restricted to 500. If this amount is exceeded, a warning message will be displayed in the beginning of list. Double-click on this message to display all items, however this operation can take very long time.

While navigating the **Project Tree**, press the right mouse button to display context popup menu. This menu, for instance, allows you to refresh the tree, open library setup windows, and open one or more selected cells. These commands are also available from the main menu category **Hierarchy→Explorer**, which is changed together with the current selected item in the **Project Tree**.

You can also open selected cells by double clicking the mouse on them.

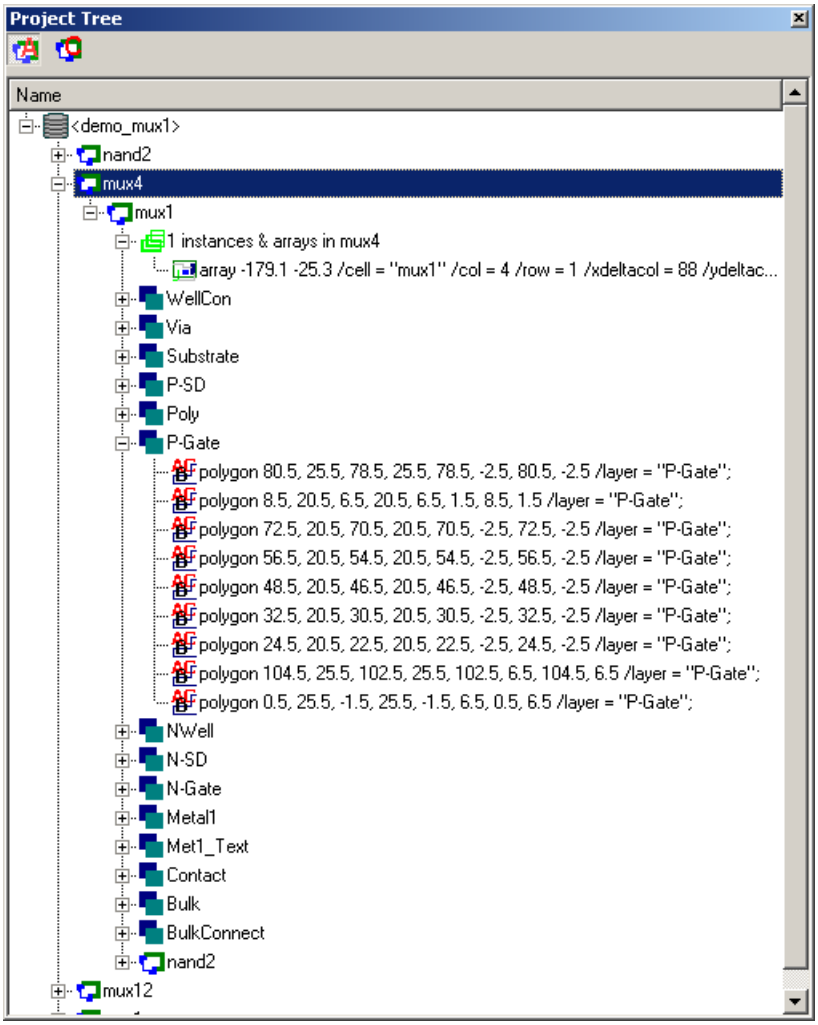

Project Tree
Name
<demo_mux1>
nand2
mux4
mux1
1 instances & arrays in mux4
array -179.1 -25.3 /cell = "mux1" /col = 4 /row = 1 /xdeltacol = 88 /ydeltac...
WellCon
Via
Substrate
P-SD
Poly
P-Gate
polygon 80.5, 25.5, 78.5, 25.5, 78.5, -2.5, 80.5, -2.5 /layer = "P-Gate";
polygon 8.5, 20.5, 6.5, 20.5, 6.5, 1.5, 8.5, 1.5 /layer = "P-Gate";
polygon 72.5, 20.5, 70.5, 20.5, 70.5, -2.5, 72.5, -2.5 /layer = "P-Gate";
polygon 56.5, 20.5, 54.5, 20.5, 54.5, -2.5, 56.5, -2.5 /layer = "P-Gate";
polygon 48.5, 20.5, 46.5, 20.5, 46.5, -2.5, 48.5, -2.5 /layer = "P-Gate";
polygon 32.5, 20.5, 30.5, 20.5, 30.5, -2.5, 32.5, -2.5 /layer = "P-Gate";
polygon 24.5, 20.5, 22.5, 20.5, 22.5, -2.5, 24.5, -2.5 /layer = "P-Gate";
polygon 104.5, 25.5, 102.5, 25.5, 102.5, 6.5, 104.5, 6.5 /layer = "P-Gate";
polygon 0.5, 25.5, -1.5, 25.5, -1.5, 6.5, 0.5, 6.5 /layer = "P-Gate";
NWell
N-SD
N-Gate
Metal1
Met1_Text
Contact
Bulk
BulkConnect
nand2
mux12

Figure 10-2: Project Tree

The **Project Tree** tree displays the following item types:

- Project/library item: Labeled by logical name of project/library.
- Cell item: Labeled by cell name.
- Instance Bundle item: A special item that further spawns items for all **instances** and **arrays** of the given cell in its **parent**. The bundle item is labeled by the number of these instances and arrays.
- Instance item: Labeled by the LISA script command which creates this instance. This command includes instanced cell name, coordinates, etc.
- Array item: An array is also marked by the text of LISA script command.
- Layer item: Labeled by name of layer.
- Graphics primitive item: Labeled by the LISA script command which creates this instance.

Expanding/Collapsing Tree

A **Project Tree** may be in collapsed, expanded, or partially collapsed states. If a tree item has a plus [+] button on the left side of its image, it has **children** hidden, which may be expanded by clicking the plus button. If a tree item has a minus [–] button, its descendants may be collapsed by clicking the minus button. You may also walk along the tree by using arrow keys, and expanding or collapsing branches by using the right and left arrows, respectively.

10.3: Edit-in-Place

Edit-in-Place menu (see Figure 10-1) allows you to select a cell instance at any level of hierarchy for editing, while working in the immediate surroundings of this instance in the chip.

All functions of the **Edit-In-Place** menu can be accessed through the **Edit-In-Place** toolbar. This toolbar can be displayed by clicking the **EIP** button.

Note: In EXPERTVIEWS, the **Edit-in-Place** group of commands only allows viewing, not editing.

To make the function of **Edit-in-Place** clear, the following terms are defined.

An **EIP** session is started by the menu commands **Hierarchy→Edit-in-Place→Enter into Subcell**, or **Hierarchy→Edit-in-Place→Direct Pick**, and is ended by the menu command, **Hierarchy→Edit-in-Place→Exit**. **Edit-in-Place→Enter into Subcell** may need to be clicked several times during the session, but until **Exit** is clicked, no new session begins.

During an **EIP** session, you may enter into a cell instance, then enter into an instance that belongs to the first instance, and so on, going deeper and deeper into the hierarchy. The nested chain of instances from the top of the hierarchy to the edited cell instance defined by the last **EIP→Enter** operation is called an **EIP Chain**.

10.3.1: Enter into Subcell

Purpose

The **Edit-in-Place→Enter into Subcell** command corresponds to the first button on the **EIP** toolbar. It switches the **EIP-entering** mode of editing on (just as the **Select→By Click/Drag** command switches on the **selection** mode of editing).

Execution

- Click **Hierarchy→Edit-in-Place→Enter into Subcell** or the corresponding **EIP** toolbar button to select this operation.

The cursor assumes the **EIP** shape indicating that you can now enter into any instance belonging to the **edited cell** by pointing-and-clicking in order to edit the cell of this instance in place.

Now you may select a cell instance which you want to edit in place. You may repeat entering into instances deeper and deeper until you select some other editing tool.

Note: **EIP** mode is sensitive to the options activated in **Select** menu. If the option **Select→Ignore Instances** is active, it will not be possible to EIP any cells.

- Point the cursor at a cell instance belonging to the currently edited cell and click the left mouse button. The **edited cell** is defined to be:
  - Either specified by the last **Enter/Up/Pick** operation as described below,
  - Or, if the **EIP** session was just started, it is the **active cell**.
- If the **edited cell** is not the **active cell**, and if there are any changes in the **edited cell**, the **End Editing** dialog appears, asking you what to do with the changed cell/instance. It is described in Section 10.3.5: "End Editing Dialog".

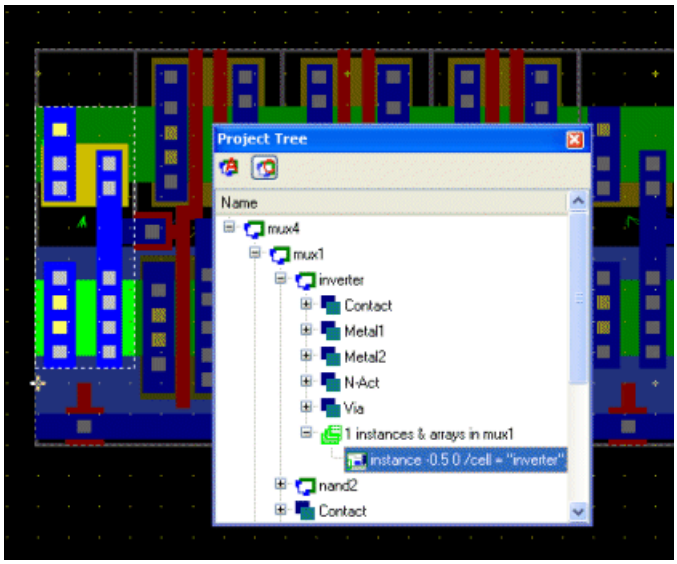


Figure 10-3: EIP at a Depth of Two

- The cell that corresponds to the clicked instance becomes the **edited cell**. It may be edited in the same way as the **active cell**, the difference being that you work within a specified instance of the edited cell, seeing the whole layout as its surrounding.

  The surrounding cannot be edited at the same time. To do this, you must apply the **EIP→Up** operation (see below) to go higher in the hierarchy to the parent instances. After editing parent instances, use the **EIP→Enter into Subcell** sequence to go back down.
- This operation changes the **EIP-chain**; now it becomes the nested chain of all instances that contain the current edited cell instance, in other words, the chain of all predecessors of the just entered instance in chip's hierarchy tree.

If the **Edit-in-Place→Dim** or **Edit-in-Place→Hide** settings were used, the appearances of the edited cell and its surroundings differs as described below.

Figure 10-3 shows in a current cell called `mux4` an **EIP** at a depth of two. This **EIP** is the result of first entering into a `mux1` instance which is the `(0,0)th` item of an instance array and then entering into an instance of inverter in this `mux1` instance. The surrounding is dimmed. One may edit the inverter

cell by all available tools. The corresponding hierarchical tree window can be activated with the menu **Hierarchy→Explorer→Project Tree**.

10.3.2: Direct Pick of Object in Hierarchy

The **Hierarchy→EIP→Direct Pick** command allows you to pick an object for editing in place anywhere in the cell hierarchy. To facilitate the choosing of the required object, preselection highlighting is used. When you click at a primitive object, you will enter into the cell instance containing this object anywhere in the hierarchy for editing in place.

"Direct Pick" EIP obeys the settings: **Select→Ignore instances** and **Select→Ignore primitives**.

The message line shows the cell name for the picked object.

10.3.3: Up

The **Edit-in-Place→Up** command corresponds to the third button of the **EIP** toolbar.

This action is applicable only if the current **edited cell** is not the current active cell. Otherwise, the corresponding menu item and button will be inactive (grayed).

Execution

- Click on the **Hierarchy→Edit-in-Place→Up** menu or the corresponding **EIP** toolbar button.

Results

- If there are any changes in the **edited cell**, the **End Editing** dialog appears, questioning what to do with the changed cell/instance. It is described in Section 10.3.5: "End Editing Dialog".
- The instance that contains the current edited cell instance becomes the new edited cell instance, in other words, it moves up the **EIP chain** in chip's hierarchy.

This action does not change the **EIP chain**. The items on the buttons remain available for the **Enter into Subcell** operation.

If the **Show-in-Place→Dim** or **Show-in-Place→Hide** settings were used, the appearances of the edited cell and the surrounding differs as described below.

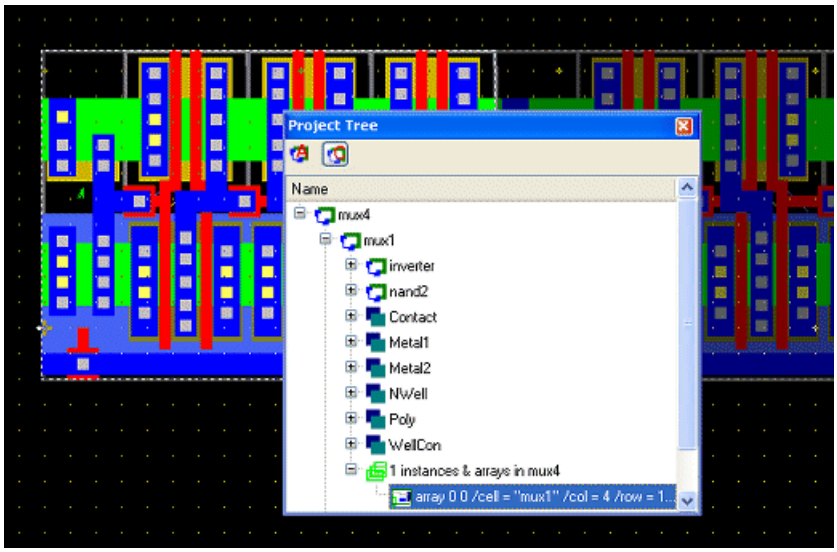


Figure 10-4: EIP→Up applied to layout from Figure 10-3

10.3.4: Exit

Purpose

The **Edit-in-Place→Exit** command corresponds to the fourth button of the **EIP** toolbar. It terminates the EIP session.

Execution

- Click on the menu command, **Hierarchy→Edit-in-Place→Exit** or the corresponding **EIP** toolbar button.

† Results

- If the **edited cell** is not the **active cell**, and if there are any changes in the **edited cell**, then the **End Editing** dialog appears, questioning what to do with the changed cell/instance. It is described in Section 10.3.5: "End Editing Dialog".
- If there are no changes, the **EIP session** is ended.
- Pressing **Cancel** in the **End Editing** dialog box returns you to the **EIP session**.
- Pressing **OK** confirms the selected changes (see Section 10.3.5: "End Editing Dialog") of the chip layout, and the **EIP session** is ended.
- At the end of the **EIP session**, the **EIP chain** is reset.

10.3.5: End Editing Dialog

Processing of all EIP commands (**Enter into Subcell**, **Up**, **Direct pick**, **Exit**, etc.) begins from this dialog (Figure 10-5), if the **edited cell** is not the **active cell**, and there are any changes in the **edited cell**. Editing of the active cell is processed as usual.

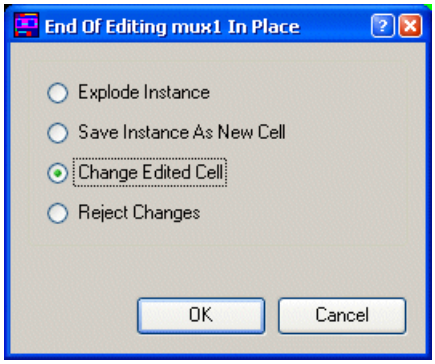


Figure 10-5: End Editing dialog box

There are several ways of introducing the performed changes made into the design as shown by the radio buttons of the dialog box:

Execution

- Select one of the following options by clicking the corresponding radio button:
  - Explode Instance
  - Save Instance As New Cell
  - Change Edited Cell
  - Reject Changes

  (The options are described in detail below.)
- Pressing **Cancel** closes the dialog box and returns you back to whatever was being edited.
- If you press **OK,** changes of the layout will proceed according to the selected option, as described below.

End of Editing Options

- **Explode Instance:** The edited instance is exploded in its parent cell.
- **Save Instance As New Cell:** Changes in the edited cell are applied only to the current instance. EXPERT creates a new cell whose instance is the just edited instance. Therefore, the **Cell Name Selection Form** titled "**Choose New Name**" appears with the following fields.
  - **New Cell Name:** A name of the new cell must be entered into this field. Initially this field contains the name of the **edited cell** slightly changed to make it unique.
  - **List of Cells:** The list of all cell names existing in the project. Double-clicking on a name from this list copies this name into the **New Cell Name** field. You may further edit it to produce a new name.
  - **Cancel:** Abandons the creation of a new cell and returns to the **End Editing** dialog.
  - **OK:** Pressing this button after cell name selection, completes the operation.
- **Change Edited Cell:** Changes in the edited cell are applied to all its instances (as if the cell were edited normally, in its own window.) This is the default choice.
- **Reject Changes:** The currently edited cell restores its layout to what it was before it was edited.

If the **OK** button was pressed, two outcomes are possible:

- The message, "The project already has such a cell name!" appears. You must click the **OK** button of the message box and repeat name selection.
- **Choose New Name** dialog box closes. The operation was performed successfully. **End of Editing** dialog box also closes.

Note: If you edit an instance of an array and exit from EIP by using the option, "Save Instance As New Cell", the array as a whole will be changed, i.e., its instanced cell will be changed.

10.3.6: EIP Highlighting

The following commands of the **EIP** submenu determines how the edited cell instance and the rest will visually differ under **Edit-in-Place**.

- **Equalize:** No difference in appearance of the edited cell instance and the rest. This command corresponds to the fifth button on the **EIP** toolbar.
- **Dim:** The surroundings are dimmed with respect to the edited cell.
- **Hide:** Only the edited cell instance remains visible.

10.4: Create Cell in-Place

Purpose

Create a new cell from the set of selected layout objects, and replace the selected set by the new cell.

Execution

- Select a set of layout objects to lump together into a new cell.
- Click on the **Hierarchy→Create Cell in-Place** menu. The **Cell Name Selection Form** titled "**Create Cell in Place**" can be seen with the following fields:
  - **New Cell Name:** You must enter a name for the new cell into this field. Initially this field contains a proposed default name. It will be used if another name is not entered.
  - **List of Cells:** This is the list of all cell names existing in the project. Double clicking on a name from this list causes it to appear in the **New Cell Name** field. It may be further edited to produce a new name.
  - **Cancel:** This button aborts the **Create Cell-in-Place** operation.
  - **OK:** Press this button after selecting the cell name to complete the operation.

If the **OK** button is pressed, there are two possibilities:

- A message that says "The project already has such a cell name!" appears. You must click the OK button of the message box and repeat name selection.
- **Create Cell in Place** dialog closes. The operation was performed successfully.

Results

This operation changes only the **hierarchy** of the active cell; it does not change its geometry (i.e., the **flattened** active cell remains the same).

If the operation succeeds, the project changes as follows:

- A new cell is added to the project. It is created from the selected objects of the active cell. The origin of the new cell is defined as follows:
  - If a reference point is active, then the origin of the created cell is set to this reference point.
  - In particular, one must set the reference point to (0,0) (and activate it), if one wants the selected objects to have the same coordinates both in the old and new cell.
  - If no reference point is active, then the origin of the new cell is the lower left corner rounded down to the minimal grid.
- The coordinates of the objects in the new cell are obtained by subtraction of the coordinates of the new cell origin described above in the active cell, from the selected object's coordinates in the active cell.
- The instance of the new cell is added to the active cell. It is positioned at the lower left corner of the bounding box of the selected objects.
- The selected layout objects are deleted from the active cell.
- No objects remain selected in the active cell.

See also: **Cell→Create**, **Select→Save Selected→As Cell...**

10.5: Explode Selected

Purpose

Explode the selected instances and arrays, in other words, move the contents of selected instances/ arrays up one level of the cell hierarchy.

Execution

- Select instances/arrays to explode.
- Click on the menu command, **Hierarchy→Explode Selected**.

Results

- The operation does not affect any objects of the active cell other than cell instances and arrays.
- Only the selected cell instances and arrays are affected by this operation.
- All layout objects from the selected instances/arrays become objects of the active cell while preserving their absolute placement in the design.
- The instances/arrays that were selected in the active cell are removed from it.

See also: **Hierarchy→Explode All**, **Hierarchy→Flatten**

10.6: Explode All

Purpose

Explode all instances and arrays in the active cell, in other words, move the contents of all instances/ arrays up one level of the cell hierarchy. This operation is equivalent to the **Select→All + Hierarchy→Explode Selected** sequence**.**

Execution

- Click on the menu command, **Hierarchy→Explode All**.

Results

- The operation does not affect any objects of the active cell other than cell instances and arrays.
- All layout objects from all instances/arrays in the active cell become objects of the active cell while preserving their absolute placement in the design.
- The instances/arrays that were in the active cell before the explosion are removed from it.

See also: **Hierarchy→Explode Selected**, **Hierarchy→Flatten**

10.7: Flatten

Purpose

Flatten the active cell, in other words, completely remove the hierarchy.

Execution

- Click on the menu command, **Hierarchy→Flatten**.
- Select file name for the flattened database at the opened file selector window. Only files in `.eld` format may be produced by this operation.
- Click the **Save** button in the file selector window.

Result

The `.eld` file with the selected name is created containing the flattened geometry of the cell.

See also: **Hierarchy→Explode All**, **Hierarchy→Selected**

10.8: Replacing Cell Instances

The menu command **Hierarchy→Replace Instances→In Current Cell...** replaces all instances of one cell by instances of another cell everywhere in the active cell.

Finally, the command **Hierarchy→Replace Instances→In Current Project...** replaces all instances of one cell by instances of another cell everywhere in the current project.

The command **Hierarchy→Replace Instances→Selected Instances...** replaces all selected instances by instances of another cell. Unlike the other two operations, instances of different cells may be replaced by instances of the same cell.

This page is intentionally left blank.

Chapter 11: Setup

11.1: Setting Up Expert&ExpertViews

The **Setup** submenu (Figure 11-1) allows you to set various options and parameters in EXPERT, namely:

- technology setup, including:
  - layer setup (including default layer drawing order)
  - layer connectivity setup
  - device setup
  - grid setup
  - general technology-related settings
- import/export setup for GDSII, CIF and Applicon formats
- ruler setup
- gravity setup
- DRC script run
- bindkey (keyboard shortcut), custom menu, and custom colors setup
- autosave setup
- viewing and editing options setups

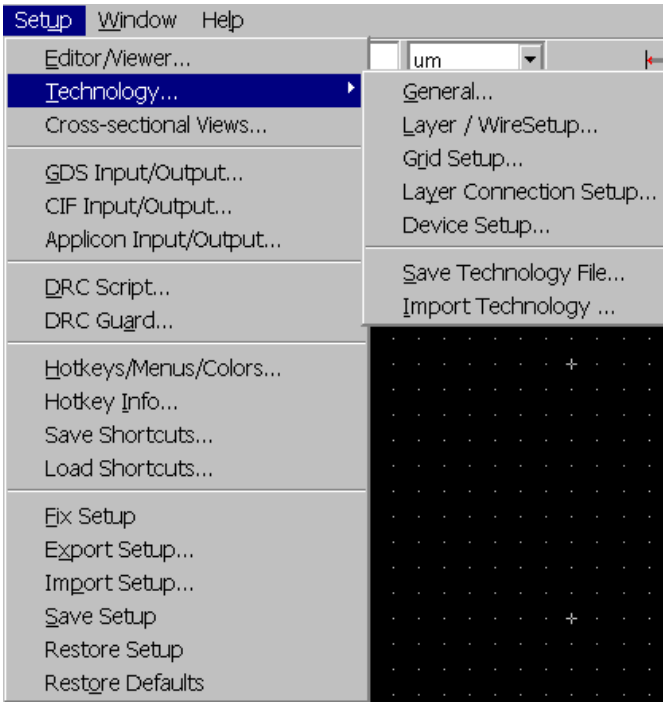


Figure 11-1: Expert Setup Submenu

All the panels which can be accessed with the **Setup** menu are gathered in the same window, the **Preferences** window.

In addition, a number of operations have their own modes to be set up before execution. These modes are found in the corresponding menus.

11.2: Managing Setups

In EXPERT, all settings may be roughly divided into two main groups: technology settings and editor/viewer preferences.

11.2.1: Technology Settings

Technology settings are inherently related to the design itself. They are stored in the corresponding technology file with a `*.tcn` extension, and in the project file itself, if it is in EXPERT'S internal format (`*.eld`). See Chapter 2: "Basic Notions", Section 2.5.8: "Technology files (*.tcn file)" for more information.

The following settings are stored in the technology file:

- user-definable technology ID string
- measurement units
- common grid setup
- layer definitions (including default wire styles, individual grid settings and derivation statements)
- layer connectivity info (see chapter 3.4.2 in Simucad's GUARDIAN USER'S MANUAL)
- physical parameters of layers
- device definitions (see chapter 3.4.3 in Simucad's GUARDIAN USER'S MANUAL)

There are several ways to create a technology file:

1. Create the required settings from EXPERT'S user interface (described in this chapter), and then save them selecting **Setup→Technology→Save Technology File...**.
2. Import from Dracula's runset as described below. This possibility is useful when you are converting Dracula DRC decks.
3. Import from foreign technology/display definition files (**Setup→Technology→Import Technology...**).
4. Edit it manually using a text editor.

Generation of Technology File from Dracula DRC Runset

You can create a technology file based on the `INPUT_LAYER` block of Dracula DRC deck. This is done as follows:

- In the DRC script panel (menu **Verification→DRC→DRC Script Panel**), open the **DRC Import Setup** window (menu **Setup→DRC Import Setup**). Select the **Generate new technology file...** option;
- Load the Dracula DRC deck from the **Import** command (menu **File→Import**).
- You are prompted to select a file name. If the file name `<FName>` is selected, three files are saved:
  - `<FName>.dsf` – is the converted script.
  - `<FName>.log` – is the conversion log.
  - `<FName>.tcn` – is the corresponding technology file.

Note: The technology file imported from an INPUT_LAYER block contains minimal data necessary to run the converted DRC script.

Import from Foreign Technology/Display/Netlist Definition Files

Selecting **Setup→Technology→Import Technology...** open the Technology Import dialog. This dialog has two pages: **Display** and **Connectivity/Devices**.

The **Display** page is used to create EXPERT'S technology, color library and stipple library files from the Cadence technology file dump and display the resource file. To obtain the Cadence technology file dump, select **Virtuoso→Technology File→Dump** with the setting **All classes** set, and type in the ASCII technology file name. The **Display resource file** has a `*.drf` extension. Sometimes, display resources can be defined in the ASCII technology file itself.

The **Connectivity/Devices** page is used to generate device and netlist definitions for HIPEX-NET netlist extractor from Dracula decks containing this information. At this page, the input file is Dracula's deck.

- The **Layer Definitions** field identifies EXPERT'S technology file, which contains converted statements of derived layers.
- The **Connectivity** field identifies EXPERT'S technology file, which contains converted connectivity definitions.
- The **Devices** field identifies EXPERT'S technology file, which contains converted device definitions.
- The **DRC Script for Layer Generation** field identifies GUARDIAN DRC'S script file, which contains converted statements for derived layers used for device and connectivity definition.
- The **DRC Script for DRC Checking** field identifies GUARDIAN DRC'S script file, which contains converted statements for DRC checks, which can be in the same input DRACULA deck and use connectivity statements for connectivity-based DRC checks.

Conversion into any of the destination files can be skipped by removing the check mark in the corresponding checkbox. Any of the three technology fields may refer to the same technology file.

11.2.2: Security of Technology Settings

What is Protected

The **TCN Security** mode prevents project technology information from being inadvertently changed. If this mode is on, then the operations listed below are deactivated:

Setup→Technology→Save Technology File...;

- minimal and snap grid settings;
- layer connectivity settings;
- layer definitions (name, GDSII number/datatype, wire styles, etc.), except for color and stipple patterns;
- color library selection and modification;
- editing of stipple patterns.

How to Set the TCN Security Mode

To switch this mode on, do the following:

1. Create an EXPERT configuration file `tcnsecur.ecf` (or some other name) with the following two lines:

```
[Application]
TCNSecurity=1
```

2. Load it by selecting **Setup→Import Setup...**.
3. Save settings by selecting **Setup→Save Setup**.

Following these actions, the user of the current PC will be working with EXPERT in `TCN` security mode.

To switch this mode off, do the same as above only using an `ecf` file with the following two lines:

```
[Application]
TCNSecurity=0
```

This method of managing security is not intended to prevent deliberate actions. This protection will be more flexible in subsequent releases, t.

11.2.3: Editor Preferences

These settings control various viewing, drawing, loading/saving, and checking options. They are stored in the permanent storage (the user section of registry for Windows, or the hidden directory in the user's home directiry for Linux/Solaris). You may export/import these settings to/from text files with a `*.ecf` extension from the **Setup** menu. The path to the most recently loaded `ecf` file is remembered in EXPERT settings. To open this folder, select **Setup→Export Setup...** and **Setup→Import Setup...**.

There are two ways of loading settings from a `*.ecf` file:

1. Select **Setup→Import Setup...**
2. Start EXPERT with the command-line switch `/ecf:<filename.ecf>`.

For example, you can create a shortcut with the following "Target" command line:

```
C:\Silvaco\etc\GuiAppStarter.exe
       -lib-dir-name Expert  -exe-name Expert
       -V 3.2.18.R -ecf:f:/aa.ecf
```

If you use this shortcut, EXPERT will load `<filename.ecf>` upon startup.

A number of user interface settings may be assigned to the current window, such as grid style, snap modes, selection modes, hierarchy viewing mode, and shortcut key assignments. If you change some of them, normally the changes are valid only for the current window. If you open a cell in a new window, apply default values to it. There are three ways of changing these defaults as described below.

The following group of commands from the **Setup** submenu allow you to transfer editing setups between windows and EXPERT's sessions.

- **Fix Setup:** By this operation, the settings of the active window are set as default or as **active** for your system. This means that all newly opened windows assume these settings until you explicitly change them. Already opened windows retain their settings. Upon normal termination, EXPERT saves the last fixed setup into the Windows registry, so that it will be used when EXPERT starts next time, unless it is overridden by the `/ecf` command-line switch described in this section.
- **Export Setup:** The settings of the current setup are stored in a file for later use.
- **Import Setup:** The previously saved settings are read from a file and applied to the rest of the current session.
- **Save Setup:** The settings of the active session are written into the permanent storage of the current user.
- **Restore Setup:** The settings are read from the permanent storage and applied to the rest of the current session.
- **Restore defaults:** The settings predefined in EXPERT are applied to the rest of the current session.

Hint: When working on multiple projects, it may be useful to load setup from a file upon starting an EXPERT application using the "Autorun script" feature of the menu command, **Setup→Editor/Viewer...→Editing**.

Custom keyboard shortcuts, custom menus and custom colors are saved/loaded by their own commands: **Setup→Shortcuts→Save Shortcuts...** and **Setup→Shortcuts→Load Shortcuts...** (see Section 11.9: "Shortcut, Custom Menu, Custom Color Setups" for more information).

Note: Another way to store editor preference in a separate file is to open the **Manage Preferences** panel of the editor Preferences window. This can be done with the menu **Setup→Setup**, then activate the **Manage Preferences** panel. The **Import...** and **Export...** buttons of this panel correspond to the Import setup and Export setup commands described above. The **Factory settings** button is equivalent to the **Restore defaults** command.

11.3: Editor/Viewer Form

The **Setup→Editor/Viewer...** menu command opens a form with several pages that control various aspects of editing/viewing the layout.

11.3.1: General Page

- Lock database units: If this option is checked, you cannot modify the values of database units. This is a safety option to maintain the same database unit for all libraries of the current design.
- Database unit for new/imported projects, um and
- Current database unit, um:

The size of the internal measurement unit for a newly created `.eld` file and for an existing loaded `.eld` file, respectively.

Since all coordinates in EXPERT are internally represented as integers, the value of this parameter affects:

- precision of data representation;
- maximal size of the layout.

You may select four values from the drop-down list: `0.0001um`, `0.001um`, `0.01um`, and `0.1um`. The first value, 0.0001um, corresponds to the database unit of some older versions of EXPERT.

If you want to use a new value of the database unit, you must change it **before** loading from GDSII, or other external formats, or **after** loading from EXPERT'S (ELD) format.

Warning: When you change this value in an open project, all coordinate values will be recalculated in the current project. It has two effects:
– This recalculation may take a noticeable amount of time for large projects.
– If there were libraries linked to the project, they may become invalid if they have an old value of database unit. In the current version of EXPERT, the current project and all its libraries must have the same value of the database unit.

- **Measurement unit:** The unit of measurement may be selected from the drop-down list. It can be one of `um`, `nm`, `mm`, `cm`, `m`, `mil`, `inch` and `Lambda` custom unit of measurement. If **Lambda** unit is selected, you have the option of setting its value in microns in the **Measurement unit, in um** field (it is activated if Lambda unit is selected).
- **Load maximum:** Sets the maximum amount of physical memory, in megabytes, for storing layout objects. Zero value means some default value. It is recommended that you change this value in heavy multitasking sessions only.
- **Default type of project for load/save file:** You may choose one of "GDSII", "EXPERT", "CIF", "Applicon", or "Recent". The latter selection means that EXPERT will remember the last opened/saved project type. In all remaining cases, EXPERT will start **Open** and **Save As** dialogs with the selected project file type.
- **Idle time for SFLM, min:** Recall, SFLM is SIMUCAD FLOATING LICENCE MANAGEMENT. If a copy of EXPERT is idle longer than the allotted time, the license becomes available to the next user starting EXPERT. The copy of EXPERT that lost its license still has options to save data and exit or try to

regain the license. The corresponding dialog will show up when EXPERT resumes its activity. Zero value for idle time means that the license is released only when the application terminates.

- **Disallow saving into GDSII, CIF, Applicon formats:** If this option is checked, then the command **Project→Save As...** will give the only option to save into the `eld` format. This option provides an extra protection of inspected layouts.
- **Write scratch layers into Expert (ELD) files:** Allows to save scratch layers as **scratch** layers in ELD files.
- **Write scratch layers into GDSII files:** Allows to save scratch layers as *ordinary* layers in GDSII files. This option is duplicated in **Setup→GDS Input/Output...→GDS Output** option **Write scratch layers**.

11.3.2: Viewing Page

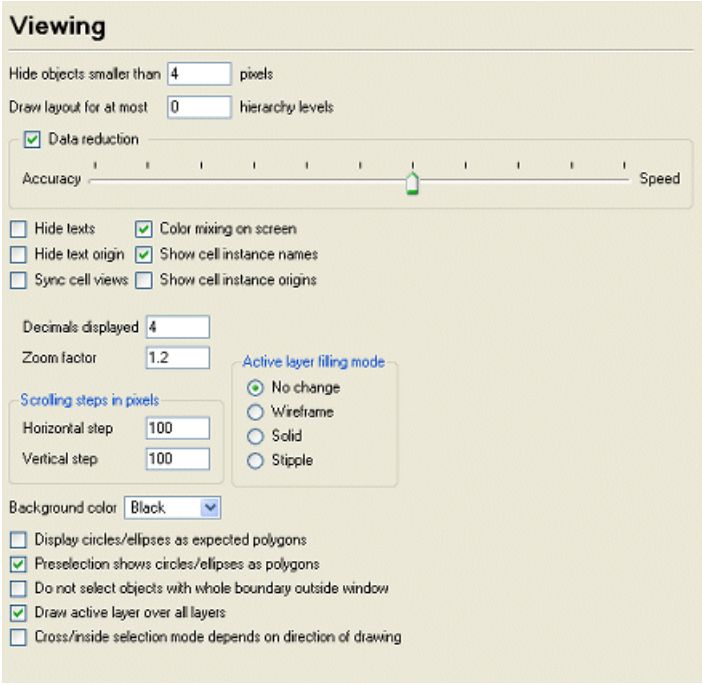


Figure 11-2: Viewing Page

- **Hide objects less than [N] pixels:** This option improves screen drawing speed by not drawing very small objects. If zero [N=0] value is set for pixels, all objects are drawn on the screen.
- **Draw layout for at most [N] hierarchy levels:** When a fixed-depth viewing mode is on, say, with depth=D, then only hierarchy levels D, D-1, ..., D-N are drawn. If N=0, then all levels from 0 to D are drawn.
- **Hide texts:** Text will not be drawn and will be nonselectable.
- **Hide text origin:** Text origin marks will not be drawn.
- **Sync cell views:** If this setting is checked, then all views of the same cell are redrawn simultaneously if any of them is edited.
- **Color mixing on screen:** Acts on the drawing in solid mode and stipple mode on the screen. It means that when drawing on the screen in these modes, the colors of overlapping pixels for objects from different layers will mix.

If the Color mixing option is off:

- In the stipple mode of display, the **fill** pixels of stipple patterns are visible in the order of layer drawing, and pixels of lower layers are visible through **holes** in stipple patterns of the upper layers.
- In the solid mode of display, the color of areas where shapes are overlapping will be the color of the layer which is the first of all in the current layer order in the layer bar. The active layer can be over all layers independently of its place in the layer order if the specific option **draw active layer over all layers** is active in the panel.

Drawing is faster in color mixing mode. The default setting for **Color mixing** is **on**.

The colors on plot and on screen in stipple mode for heavily overlapped drawings will look the same if you set **Color mixing** mode in the **Setup→Editor/Viewer...→Viewing**, and color mixing in mask mode in the "Miscellaneous" plotting style setup.

Note: When activating the Color mixing in solid mode, some areas of shapes can appear as white when too many layers are overlapping at these places (for example, after derived layers generation). Therefore, it can be useful to hide unused layers in order to avoid these white areas.

Warning: Some plotters do not support color mixing in "Direct output" mode ("Miscellaneous" plotting style settings).

- **Show cell instance names:** When instances are drawn as frames (e.g, in lazy view mode, their cell names are written as well).
- **Show cell instance origins:** Cell instance reference points are drawn as small markers.
- **Decimals displayed**: The number of digits displayed after the decimal point in outputs.
- **Zoom Factor**: The amount of zooming/unzooming performed when you run **View→Zoom→Zoom In** or **View→Zoom→Zoom Out** menu commands; press Gray Plus or Gray Minus on the keyboard, or click the plus or minus buttons at the **Command bar**.
- **Scrolling Steps in Pixels:** These two fields (horizontal and vertical steps) allow you to set the distance the screen scrolls when a scrolling arrow or a scrolling shortcut is pressed.
- **Active layer filling mode:** one of four values may be selected: wireframe, solid, stipple, and no change. This setting defines the filling mode of the active layer. When the layer is activated, its filling is controlled by this setting. When the layer is deactivated, its filling mode is restored to the previous one. For example, it is sometimes convenient to see all layers in wireframe, and the active layer is always highlighted by being drawn filled. This setting is duplicated in the **View→Filling →Active Layer Mode** submenu.
- **Background Color:** General setup allows you to select either black or white background color. In the case of a white background, all auxiliary markers will be inverted from white to black.
- **Display circles/ellipses as expected polygons**, and **Preselection shows circles/ellipses as polygons:** These options allow you to see the polygonal shape of a circle/ellipse according to the settings in this setup panel, rather than the perfect circle/ellipse.
- **Do not select objects with whole boundary outside window:** This option acts if **Select→Selection Modes→By Solid** mode is active. It excludes from selection, by box or by click, the objects whose boundaries are completely outside the part of the layout visible on the screen.
- **Draw active layer over all layers:** If it is turned on, all shapes in active layer are drawn over the shapes in other layers.
- **Cross/inside selection mode depends on direction of drawing:** If the selection frame is drawn from left to right, the **Cross** or **Inside** mode, set by **Select→Selection Modes** submenu, is

used. Contrary, if the selection frame is drawn from right to left, the mode is opposite to the option set by **Select→Selection Modes** submenu.

11.3.3: Editing Page

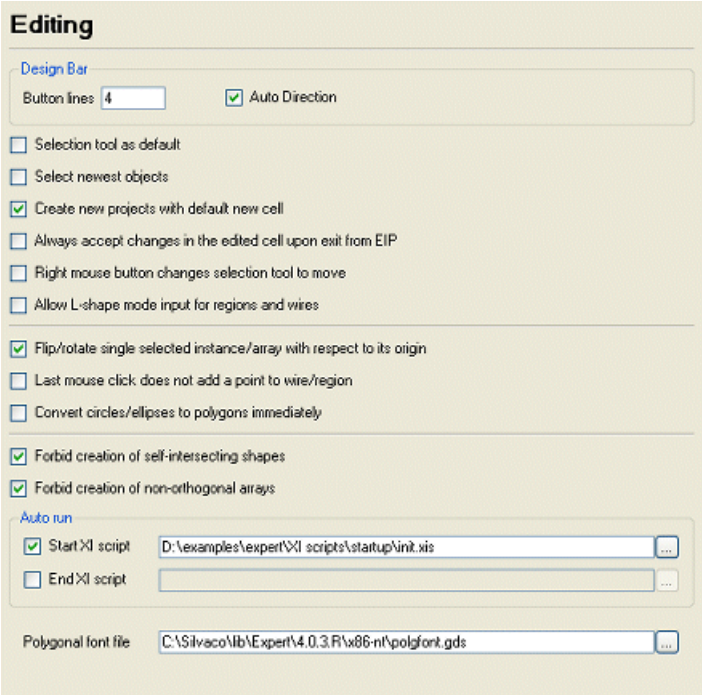

Editing
Design Bar
Button lines 4
Auto Direction
Selection tool as default
Select newest objects
Create new projects with default new cell
Always accept changes in the edited cell upon exit from EIP
Right mouse button changes selection tool to move
Allow L-shape mode input for regions and wires
Flip/rotate single selected instance/array with respect to its origin
Last mouse click does not add a point to wire/region
Convert circles/ellipses to polygons immediately
Forbid creation of self-intersecting shapes
Forbid creation of non-orthogonal arrays
Auto run
Start XI script
D:\examples\expert\XI scripts\startup\init.xis
End XI script
Polygonal font file
C:\Silvaco\lib\Expert\4.0.3.R\x86-nt\polgfont.gds

Figure 11-3: Editing Page

- **Design bar:** In this field you can specify the number of button lines of the design bar, and the orientation of the bar due to the **Auto Direction** box. With the **Auto Direction** box checked, the orientation of the **Design** toolbar will change depending where it is docked. Without **Auto Direction**, the toolbar orientation will not change in any case. It is the number of lines which will force the orientation (e.g., define 16 lines if you want the **Design** bar to always be vertical with only one column).
- **Selection tool as default:** If this option is on, the **Esc** key will switch you from other tools to **Select** mode.
- **Select newest objects:** If this option is on, each time you create a new object or modify an object, it becomes selected.
- **Create new projects with default new cell:** If this option is on, then **Project→New...** command will create a new empty cell in the new project.
- **Always accept changes in the edited cell upon exit from EIP:** Under this option, you exit from EIP without the saving confirmation dialog, and changes during EIP are always accepted.
- **Right mouse button changes selection tool to move:** If this option is checked, then the right mouse button click starts the **Move** command, if an object is selected.
- **Allow L-Shape Mode Input for Regions and Wires:** This option is used to disable the Over the Point Drawing mode (see Chapter 5: "Creating Layout Objects", "Over the Point Drawing" on page 5-5 for more information).
- **Use Property bar for Numeric Input:** This option is useful when you want to choose where to display information about shapes you are designing or modifying:
  - If checking it, you will have to activate the **Property** bar (with the menu **View→Dock Windows→Property Bar** in order to see the parameters (coordinates…) of a shape while designing or modifying it.

- If not checking it, the parameters of shapes during their creation will be displayed in a separate window called the **Numeric Input Bar**, which can be activated with the menu **Edit→Numeric Input**.

The **Property Bar** will be used only for the display of parameters of existing selected shapes.

The **Numeric Input** menu item will exist only if this option is checked. You need to restart EXPERT after changing this option in order to activate it.

- **Flip/rotate single selected instance/array with respect to its origin:** This option acts for rotate-90 and flip operations. If it is not checked, the operation is done as usual, with respect to the geometric center.
- **Last mouse click does not add a point to wire/region:** When a wire/region is created, the right mouse click terminates the operation, but does not add a point to the shape.
- **Convert circles/ellipses to polygons immediately:** Upon drawing, circles/ellipses will be immediately converted to polygons according to the current settings from the Shapes panel of this setup.
- **Forbid creation of self-intersecting shapes:** If this option is on, then self-intersecting shapes will be forbidden during creation/modification. If the option is off, you will see the warning allowing you to accept such shapes.
- **Forbid creation of non-orthogonal arrays:** If this option is on, creation of non-orthogonal arrays is forbidden (only 1x1 array is allowed to be built). A non-orthogonal array is the one whose step in a row has a nonzero Y component, or whose step in a column has nonzero X component.
- **Autorun:** Start xi-script: You may select the name of the xi-script that automatically executes when you start EXPERT. Typically, it contains definitions of custom commands and definitions of EXPERT settings for your workspace.
- **Autorun:** End xi-script: You may select the name of the xi-script that automatically executes when you exit EXPERT. Typically, it contains clean-up and backup commands.
- **Polygonal font file:** You may select the name of the GDSII file that contains 45-angle fonts for creating polygonal texts. An example of such file is `polgfont.gds` found in the installation of EXPERT. You can copy this file and create your own polygonal shapes for letters.

11.3.4: Data Safety Page

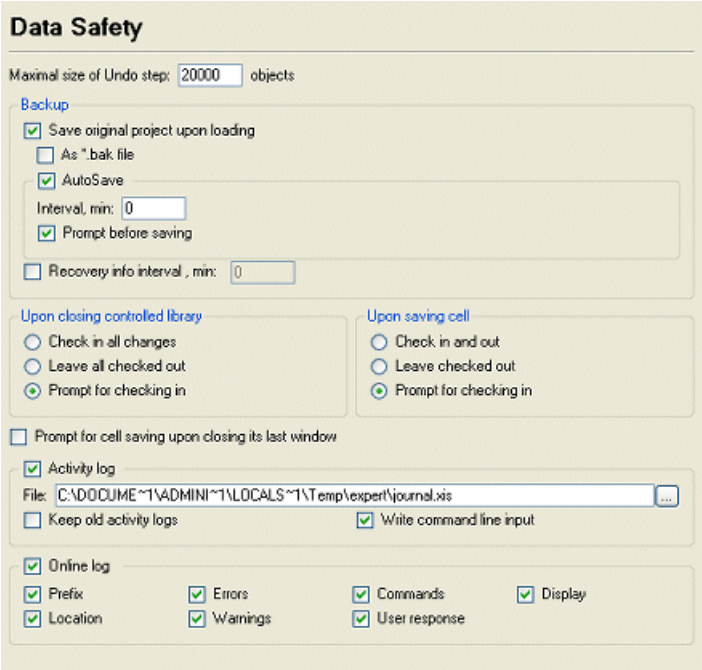


Figure 11-4: Data Safety Page

† • **Maximum size of Undo step [N] objects:** Sets the restriction on the number of objects created/changed by a **single** operation (copy, delete, etc.) to be written in the **undo** log. If an operation handles more than the given number [N] of objects, it cannot be undone. You will see the corresponding warning.

† • **Save original project upon loading:** If this box is checked, each time you load a new project, you are prompted to save a copy under a new name. If you respond by clicking **Cancel**, no backup copy will be saved.

This option has the sub-option, **As *.bak file**. If the latter one is checked, instead of asking the name for the backup file, EXPERT creates the backup file with a `.bak` extension. If the file name already exists, a new one will be created.

Note: The first way of backing up is convenient if you want to have a backup copy on another disk or even across the network, especially when the project file is large, or if you want to have several backup generations. The `*.bak` method might be convenient for smaller projects.

Note: EXPERT'S file selectors normally will not "see" files with a `.bak` extension, since EXPERT recognizes database file format by file extension only. Therefore, to open a backup project, you must rename the required `*.bak` file into a `*.eld` file using the Windows file browser.

- **Autosave interval, min:** If you enter a positive value for this parameter, the current project is autosaved each time the autosave time interval in minutes is elapsed if there were changes in the project during this period. If you enter zero, no autosave will be performed. **Prompt before saving** box activation allows you to be prompted to skip automatic save.

- **Recovery info interval, min:** The **Recovery** info is a record of all operations executed. It is written for recovery purposes if EXPERT is terminated abnormally (then you will be able to recover unsaved data). To activate the EXPERT recovery feature, check the option **Recovery info interval** and specify the interval of minutes you want to recover. EXPERT will automatically write a recovery data at the entered interval. After an abnormal termination, restart EXPERT and load the project file which may contain unsaved information. When unsaved data are detected the message box will pop up asking whether to save (or discard) recovered data, or continue to load the project with the recovered information. The last choice allows you to open and examine recovered cells (they will be marked as modified). Save or discard specific cells or whole projects by using **Save project** or **Close project**.
- **Upon Closing Shared Library:** While closing library, choose between:
  - Check in all changes
  - Leave all checked out
  - Prompt for checking in
- **Upon Saving Cell:** While saving cell, choose between:
  - Check in and out (to continue editing)
  - Leave checked out
  - Prompt for checking in
- **Prompt for cell saving upon closing its last window:** If a cell has been changed and no windows are open with this cell, choose to save changes in this cell.

† • **Write activity log:** If this option is checked, then all editing commands are stored in xi-script format in the file with the name entered into the field Activity log file. This name may be entered either by typing or by a file browser invoked by the button **"..."** next to the file name field.

Note: The user must have a license for LISA to replay activity logs.

- **Keep old activity logs:** If this option is off, then when a next project is loaded, the old activity log is overwritten. If this option is on, then the old activity logs are renamed using consecutive numbers and stored. Under this option you must periodically delete old logs manually.
- **Write command line input:** If this option is on, all the LISA commands typed in the command line will be added to the journal file.

† • **Online Log:** If this option is checked, and **Tools→XI Script→Log** window opens, the messages which are generated by xi-commands are displayed in the Log window. Message Filters for EXPERT **Log Online** panel allows you to choose the output you want to display. You can include or exclude from output the following items:
  - 'Prefix' means EXPERT output codes: 'W>' for warnings, 'E>' for errors, 'O>' for LISA DISPLAY command.
  - 'Location' filter is for adding XI script line number to the output where errors, warnings, or DISPLAY commands are appeared.
  - 'Errors' filter is to show/hide error messages.
  - 'Warnings' filter is to show/hide warning messages.
  - 'Commands' filter is to show/hide XI commands.
  - 'Display' filter is to show the result of 'DISPLAY' command.

The **Setup→Editor/Viewer...→Data Safety** dialog is shown in Figure 11-4.

11.3.5: Directories and Files Page

- Use non-default path for layout viewer/editor temporary data: If this option is not checked, then EXPERT'S temporary files are created in the EXPERT subdirectory of the Windows temporary directory (usually specified by the TMP or TEMP environment variable).

- Use non-default path for DRC temporary data and Use non-default path for netlist extractor temporary data:
  - If one of these options is checked, then the `<project_name>.DRC` (`resp.`, `<project_name>.EXTR`) working folder will be created in the indicated directory.
  - If this option is not checked, or the corresponding directory is not specified, then `<project_name>.DRC` (`resp.`, `<project_name>.EXTR`) working folder will be created in the directory containing the loaded `<project_name>.eld` file.
  - If this non-default path points to a non-existing directory, or with insufficient access rights, then DRC (extractor) will abort with the corresponding error message.
  - This option is useful when the project is stored on a remote disk, and DRC would run slowly, and if the `<project_name>.DRC` (`resp.`, `<project_name>.EXTR`) is on the remote disk in a slow or busy network.

The following fields define correspondent setting files different from the default ones:

- Color library file;
- Stipple library file;
- Plotting style file;
- Bindkey file.
- Configuration file.

11.3.6: Ruler Page

The **Ruler** page allows you to set the ruler font, color, and the style of ruler marking. Font and color may be selected from the resources provided by the OS.

- **Font:** Fonts are changed for all rulers, while all other parameters can be applied either to new rulers only or to the existing ones.
- **Color:** Color may be chosen for every ruler.
- **Graduate rulers:** If this option is on, the ruler may be set to one of three graduation options available:
  - **Auto:** It corresponds to the old way of graduation. The ruler is subdivided to make grades readable.
  - **Fixed Step:** Specify any fixed graduation step in current measurement units.
  - **Semidecimal:** A variant of auto where the marked grades are multiples of 5 multiplied by a degree of 10. Examples:
    0.21, 0.215, 0.22, 0.225, 0.23…
    1000, 1500, 2000, 2500, 3000…
  - **Endmarks only:** Display the total length of ruler segments.
  - **Apply to all rulers:** Allows you to set up the same ruler style for all rulers existing at this moment.

11.3.7: Infix Page

This page is to set **infix mode** for some editor commands that normally would require the input of two or more points by mouse click.

- Ruler Infix mode
- Zoom Infix mode
- Box Infix mode

In this mode, if a command is started by its bindkey, then the first input point is taken at the mouse position at the moment of hitting the bindkey (without mouse click). This way, the operation requires one mouse click less.

If the command is started from menu, then you still need to click the first input point.

11.3.8: Gravity Page

If the **gravity mode** is on, the cursor snaps to elements that are characteristic of the object's shape, such as vertices, midpoints, and centers.

The setup panel, **Gravity**, provides the following controls:

- **Gravity mode On:** Turn the gravity on and off.
- **Attracting elements:** Select snappable shape elements (wire centerline, wire end, wire joint, midpoint, edge, vertex, box/ellipse center, etc.).
- **Gravity distance:** Set the radius within which the gravity acts.

Gravity radius may be set in the following measurement units:

- pixels
- microns
- nanometers

Gravity acts only for the following tools:

- void tool
- create box, region, wire, ellipse, donut
- move, duplicate, modify
- paste
- ruler
- create instance

Gravity acts only for editing tools and rulers. These tools are:

- create objects (box, region, wire, ellipse, donut, instance etc)
- move, duplicate
- modify, stretch
- cut-by-line, clip in/out
- paste
- ruler

11.4: Technology Setup

11.4.1: General Page

This panel is used to choose the general settings of the technology: it belongs to the Technology setup panels group of the **Preferences** window. It can also be displayed directly from the menu **Setup→Technology→General**.

Technology ID: This is a user-defined string that identifies a particular technology file. When a new technology file is created, the default value of this string is: "Non-named technology, ver. 1.01".

Note: Since the technology data are stored inside `*.eld` files, this ID string is the only way to identify `*.eld` projects with the same technology.

Use external technology file for *.eld projects: Normally, technology data for EXPERT'S native projects is stored inside the `.eld` file. It is, however, possible to load a `.eld` project with an **external** technology file, selectable as described for loading of foreign data formats.

This is particularly useful for teamwork. It ensures that everyone uses the same technology, for example, from a file on a server.

Under this option, layers in the project and in the technology file are matched basing on:

- **Layer name** if the corresponding radio-button **Identify layers by name** is checked. In this case, the GDSII number of a layer of the technology file will replace the GDSII number of the layer with same name in the loaded `ELD` project,
- **Layer GDSII number** if the other radio-button is checked. In this case, the name of a layer of the technology file will replace the name of a layer having the same GDSII number in the loaded `ELD` project.

If the project has layers undefined in the technology file, they will be loaded anyway, but their parameters will have the same default values.

Note: You cannot replace technology for a loaded project. You must reload the project. However, you may replace colors/stipples using "layer plans".

Allow layers with the same GDSII numbers and datatypes: Supresses warnings when layers with the same GDSII parameters are created.

If this option is NOT checked, then operations **Project→New...** and **Project→Load...** that use a technology file with a pair of layers with the same GDSII numbers and datatypes will fail with the corresponding message.

A similar control is performed during technology creation/modification using **Setup→Technology→Layer / Wire Setup...**.

This control prevents unintentional data change during export into the GDSII format. Two or more EXPERT layers can be written into one GDSII layer.

If it is required to merge layers during GDSII export, it is recommended to use a layer remapping table, see **Setup→GDSII Input/Output→Output**.

11.4.2: Shapes Page

This panel is used for the setup of curved and off-grid shapes of the layout. It can be opened in the **Preferences** window (menu: **Setup→Editor/Viewer**) by selecting the General page of the Technology setup group.

† • **Polygonal Curve Approximation:** This group allows you to select the precision with which ellipses, donuts (rings) and arcs in the rolled region are approximated when converted into polygonal regions. This group contains three radio buttons and a **Value** input field. In all cases, a curve is converted into an inscribed polygon.
  - **Vertex Count:** In this case, the approximation is defined by the number of vertices into which the complete ellipse circumference is split. This means that an arc will be split into proportionally less vertices, depending on its angular size. Notice that if an arc is too small, it will be converted into a straight line segment.
  - **Angle:** An ellipse/arc will be split by vertices in such a way that the angular distance between split points does not exceed the prescribed angular value, in degrees.
  - **Deflection:** An ellipse/arc will be approximated by a polyline in such a way that the maximal deflection of the polyline from the curve will not exceed the prescribed distance, specified in current measurement units.
- **By circumscribed polygon:** If this option is on then the approximating polygon is outside the circle, i.e., it is the circumscribing one. Otherwise, the circle/ellipse is approximated by the inscribed polygon.
- **Same Y for first vertex and center:** This setting controls the conversion of circle/ellipse to polygon. If this option is checked, then the first point of the approximating polygon is taken with the same coordinate as the center of the circle/ellipse. Otherwise the polygon is created in such a way that the first side is vertical. The latter option is useful when circles are converted into octagons. You will obtain a 45-degrees octagon.
- **Offgrid wire boundary allowed:** This option affects the drawing of wires and the "Convert to polygon" operation for wires.

If this setting is off, the shape of a 45-degree-mode wire is corrected in such a way that the vertices of its boundary polygon fit on the minimal grid. As a by-product, the width of diagonal wire segments is somewhat increased. Unfortunately, this cannot be avoided.

If this setting is on, the wire shape is drawn exactly according to its centerline and (constant) width.

- **Offgrid circle/ellipse/donut boundary allowed:** In general, EXPERT tends to place all created geometry on the minimal grid if the "Snap to grid" setting is on. In some cases, it may be inconvenient. For example, sometimes a circle is converted to a polygon (by command **Tools→Polygonize→Convert All circles in project**, or when saving the project into a GDSII file). If this polygon is snapped to a grid, then subsequent operations of resizing and merging could accumulate significant deformations of the circular shape. Therefore, it is useful to force only the center and the radius of a circle to be on grid and do not snap the boundary of the corresponding polygon to grid. The corresponding option is **Offgrid ellipse/circle/donut boundary allowed**. The polygons produced under this option may be snapped to grid at the final stages of design by **Tools→Snap Selected To Grid...** and/or **Tools→Snap Whole Layout To Grid...** commands.

11.4.3: Layer/Wire Setup

Layer setup consists of two main parts: **Layer Bar** and **Layer Setup** dialog. The Layer bar is visible on screen if the **View→Dock Windows→Layer Bar** option is checked. **Layer Setup** dialog is called either from the **Setup→Technology→Layer/Wire Setup...** menu, or by clicking the right mouse button on a layer name within the **Layer Bar**. In the latter case, the layer under the cursor in **Layer bar** becomes selected (with gray background) in the **Layer Setup**, and is available for modifications there.

Settings made in the **Layer bar** apply only to the window that is active at that time. Settings made in **Layer setup** are valid for the whole project during the session and may be saved as described in Section 11.2: "Managing Setups".

Layer Bar

The **Layer** bar, shown on the right side of Figure 11-5, allows you to show/hide layers in the layout, change their visibility order, make them selectable/nonselectable, change the filling mode of a layer, and set a layer to be **active.** (Some editing operations, e.g., creation of shapes, are performed only in the **active layer**).

The **Layer** bar contains two vertical colors: **name strip** and **color strip**.

At the **Layer** bar, the name of the **active layer** is written on the colored background. Only one layer may be active at a time. The active layer is always visible. A layer is activated by clicking on its color strip.

Visible layer names appear in black, non-selectable layers have grayish names, and those layer names that are invisible appear in white.

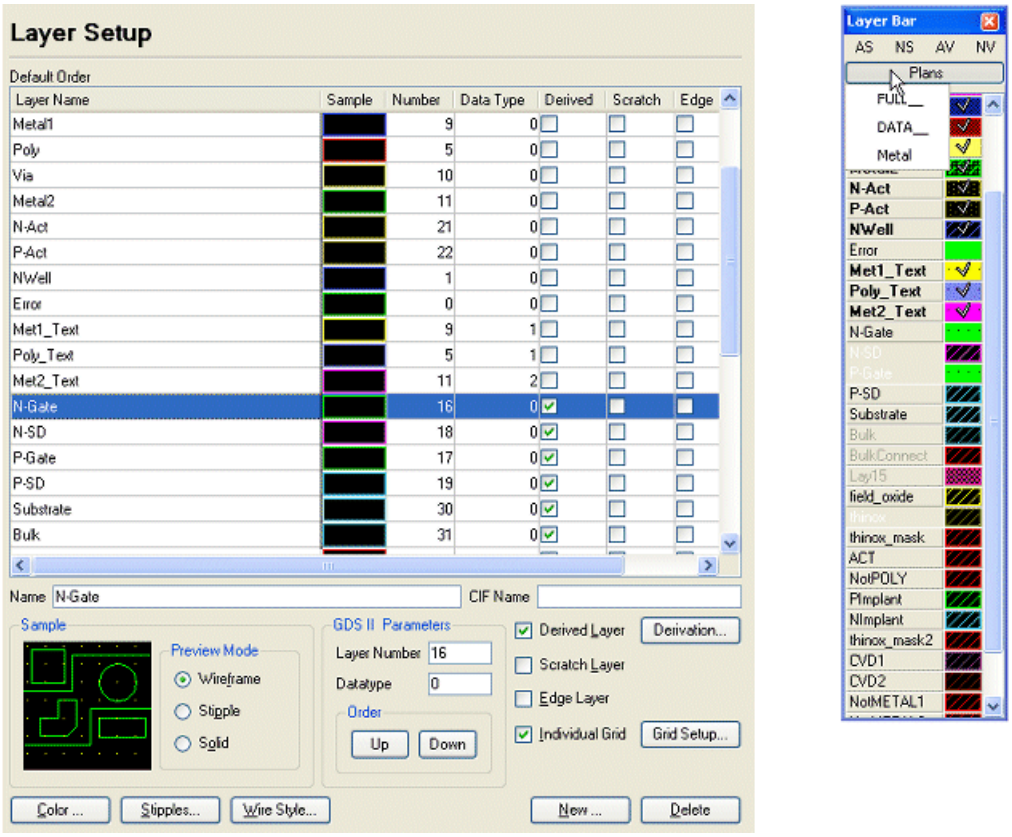


Figure 11-5: Layer Setup Dialog and Layer Bar

The **visibility** of a layer is toggled by clicking on its name strip:

- The **AV** (all visible) button makes all layers visible.
- The **NV** (none visible) button makes all layers (except the active one) invisible.

The **selectability** of a layer is toggled by clicking on its name strip while holding down the `Ctrl` button.

You may toggle selectability of all layers in the same way as it is done for the visibility of layers:

- If you click the **AS** (all selectable) button, all layers become selectable.
- If you click the **NS** (none selectable) button, all layers but the active one become nonselectable.

Layers with geometry actually present in the cell are check marked, and their names are printed in boldface.

Layer Bar Context Menu

If you right click the mouse button in the **Layer** bar (on the name strip or color strip), a contextual menu will be displayed (see Figure 11-6). This menu allows you to set several characteristics of layers:

- visibility and selectability of the pointed layer,
- visibility and selectability of all layers.
- layer setup dialog
- layer connection setup
- layers reordering modes: alphabetical, technological or GDSII-number order. (Note: custom reordering can be done in "layer/wire setup" dialog, described in Section “Layer Setup Dialog”).
- individual layer filling (wireframe, solid, stipple). This filling mode will override the common default filling mode.
- color and stipple of the pointed layer (these two items can be accessed from **Layer/wire setup** and will be described in Section “Layer Setup Dialog”).

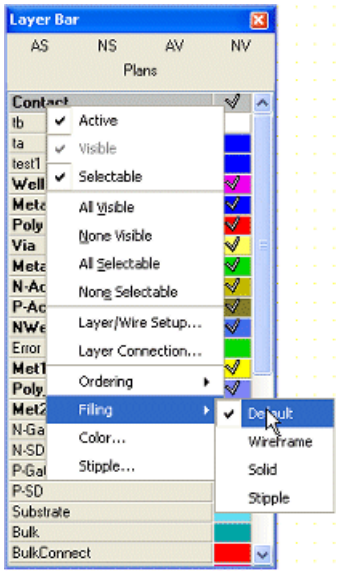


Figure 11-6: Layer Bar Contextual Menu

Layer Plans: Filling, Selectivity, Visibility, and Inclusion

At the **Layer** bar, the **Plan** command button is used to set the following properties for each layer:

- Individual filling settings
- Visible/Invisible in the layout
- Selectable/nonselectable (i.e., elements of this layer may be selectable or not). Note that invisible layers are non-selectable as well.
- Included/not included into the layer list of the **Layer** bar.

Note: You can easily filter alphabetically which layers will be included in the **Layer** bar thanks to the text field at the bottom of the **Layer** bar. Typing one or several characters in his field will display only the layer names beginning with these characters or containing these characters.

(The last feature is useful when you have too many layers, and you are tired of scrolling up and down the layer list.)

If you click on the **Plans** button in the layer bar, the **Layer Plan** panel appears as shown in Figure 11-7. The **Layer Plan** panel can also be opened with menu **View→Layer View→Layer Plan→Setup**.

Each particular combination of visibility/selectability/inclusion may be assigned a name and added to a drop-down list of **Layer Plans**. This makes it easy to switch between different layer plans.

In the **Layer Plan** panel, shown in Figure 11-7, a five-column table is used for the list of the layer names: "Layer name", "Filling", "Include", "Visible", and "Selectable".

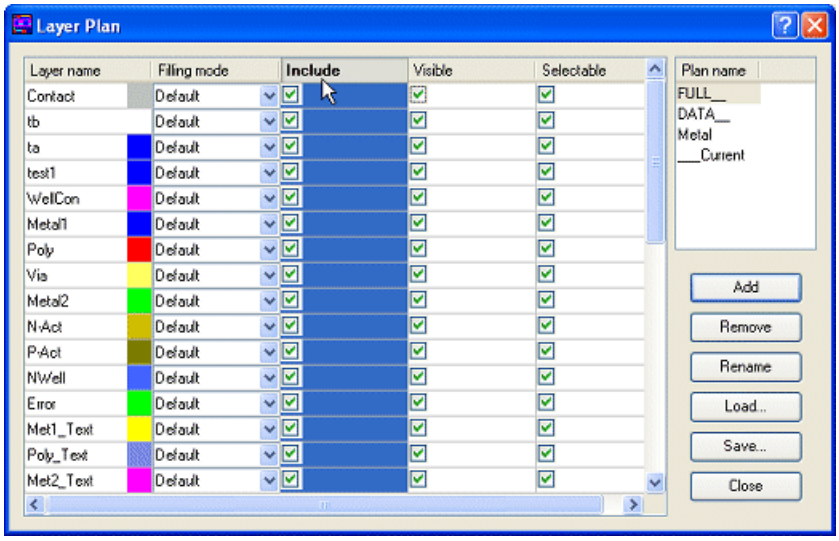


Figure 11-7: Layer Plan

Using the second column, you may set individual filling modes for layers. Select one of the four items of the drop down menu: **default**, **solid**, **wireframe**, or **stipple** by the left mouse button. The **default** filling mode means that the setting will be taken from the current common setting (**View→Filling** submenu).

The last three columns contain check marks indicating that a particular property is applicable to the corresponding layer:

- You may check on/off the layers properties individually. Changes will be applied instantaneously when closing the Layer plan panel.
- You may change all layer properties of a column by clicking on the top of the column. For example, if you click on **Include**, the whole column becomes highlighted. Then click at any checkbox in order to change the characteristics for all layers in one shot.
- You may change the property of a group of layers by clicking on the right side of the check box of several layers in the desired column, holding the Shift key depressed. Then you can check/uncheck the box of one of the selected layers: the property will be changed for the whole group of layers.

The **Layer Plan** panel contains a **Plan List** option. You can add, delete, or rename plans in it. Also, you can select any plan from the list to be used. This list can be accessed directly from the **Layer** bar.

When clicking the black arrow button on the right side of the Plans button, a drop-down list of plans appears and then any of them can be selected with the cursor.

The **Plan List** on the **Layer Plan** panel, always contains the plan "__Current", which corresponds to the one used in the **Layer** bar. The most common use of it as follows:

- You set layer visibility/selectability/filling as desired for any layer in the layer list or through the layer plan.
- In the Layer Plan panel: rename the "__Current" layer plan with the name of the layer plan you want to create (click the **Rename** button, change the layer plan name and type “return” on the keyboard).
- There are two ways to save your layer plans:
  - If you want to reuse your layer plans later in other projects, save them into separate files and load them later. For each layer plan, select it and click on the **Save** button, then you will be prompted to choose the name of the **Layer Plan** file (`*.elp`). You may load this file later in another project with equivalent technology, using the **Load** button of the **Layer Plan** panel.
  - If you want your Layer Plans to always be available in your current project, you can store them directly in a `.eld` file, together with EXPERT'S project. It consists simply of saving your project when you have defined your layer plans. Moreover you can set one of the layer plans as the default one when opening the project. You have to activate it and then execute the command **View→Layer View→Layer Plan →Fix Plan**.

Loading/Saving Layer Style (Color, Filling) using Layer Plans

Expert Layer Plan files (`*.elp`) store layer color and stipple pattern data, so that you can replace layer style without changing the technology file. This is done as follows.

- load a project.
- set your favorite color and stipple patterns.
- open the **Layer Plan** panel by clicking the **Plan** button in the **Layer** bar.
- open the **Layer Plan** panel either from menu **View→Layer View→Layer Plan**, or clicking the **Plan** button in the **Layer** bar.
- add a new plan named, e.g., `my_style-1`.
- save it. In this example, by default it will be saved in a file `my_style-1.elp`.

From now on, if you load a project with a compatible technology (i.e., with the same layer names), you may load the `my_style-1.elp` layer plan to set your layer style.

Note: When a layer plan is loaded, it does not change the appearance of the layers not listed in it.

Default “FULL__” and “DATA__” Layer Plans

When a project is loaded, EXPERT automatically creates layer plans with the names **FULL__** and **DATA__**. FULL__ layer plan shows all layers present in the technology. DATA__ layer plan shows only layers that contain geometry in the current cell.

These layer plans are automatically updated when you open a cell or select one of these plans from the layer plan drop-down list in the **Layer** bar.

Layer Setup Dialog

The **Layer Setup** dialog contains the list of layer names in default order, specified by the technology file.

You can access this panel in two ways:

- With the menu: **Setup→Technology→Layer/Wire Setup**,
- In the layer bar, right-click on a layer name and then choose **layer setup** in the contextual menu which is displayed.

This panel can be used to visualize or to modify layer parameters, first by selecting the layer in the layer list at the top of this panel, and then checking or modifying parameters which can be accessed in the fields at the bottom of the panel.

Note: When you change some layer parameters in **Layer Setup** dialog, these changes are applied to the current project only, and they are stored in the current project (`*.eld`) when saving it. These changes are never applied automatically to the original technology file, even if you loaded the project in GDSII format with an associated technology file. The only way to apply the changes to the technology file is to generate again the technology file from the current `*.eld` project with the command **Setup→Technology→Save Technology File...**.

Layer Creation

In order to add a layer in the technology, click on the **New..** button. The **New Layer** dialog box appears shown in Figure 11-8.

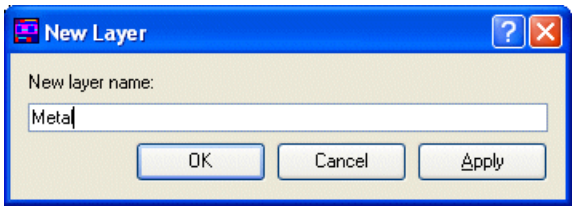


Figure 11-8: New Layer dialog box

Enter a new name then press **OK**. If you need to create several layers, click on **Apply** and repeat the sequence as many times as needed, and then press **OK**. You are returned to the main layer setup dialog so that other parameters of the new layers may be entered.

You can also specify the CIF name.

GDSII Parameters Assignment

It is mandatory to assign each layer two identification numbers which are the layer number (also called GDSII number) and the Datatype number. In the layer list, click the layer name whose settings you want to change. This name appears in the bottom **Name** field, and the layer number and datatype appear in the GDSII parameters field.

Several restrictions are checked when entering layer name and parameters:

- If a duplicate name is entered, you will be prompted to repeat input.
- If you enter a GDSII number or a datatype out of the default (0-255) range, you will be warned and allowed to change your mind.
- If you enter 2 layers with same GDSII number and Datatype, you will be prompted to change at least one of the parameters in order to differentiate the two layers. (Note: this check depends of the option **allow layers with same GDSII numbers and datatype** described in Section 11.4).

Layer order: The order of the layers in the **Layer Setup** panel is the technology order. It can be changed by moving the layer up or down in the list thanks to the corresponding buttons of the **Order** field. The new order will be saved with the project.

Layer suppression: You may delete the selected layer by pressing the **Delete** button.

Derived layer: This field concerns the layers which are defined with boolean operations between existing layers. This feature is described in Section “Derived Layers”.

Scratch layers/Edge layers: This characteristic concerns the layers which are involved in DRC scripts. They are described in Section “Scratch Layers” and Section “Edge Layers”.

Individual Grid: If this box is checked, individual grid settings will be used for this layer.

Grid Setup...: This button invokes the standard grid setup dialog, described in Section 11.4.6: “Grid Setup”, with the only difference being that the selected grid settings will be applied to the layer in question only.

Restrictions for Layer Names

Layer names cannot use those characters that are forbidden in file names.

The additional list of restrictions is:

- Layer name length must be at most 255 characters
- The following characters are forbidden in layer names:
  - All non-printable characters (with codes 0-31 and above 126)
  - =();",
  - Space character
- The first character cannot be '&' (it is reserved for marking temporary layers in the DRC script language).

Layer Color Setup

To change layer color, click the **Color** button. The **Select color** panel appears (see Figure 11-9). Either predefined colors may be selected, or a new one of your own may be constructed.

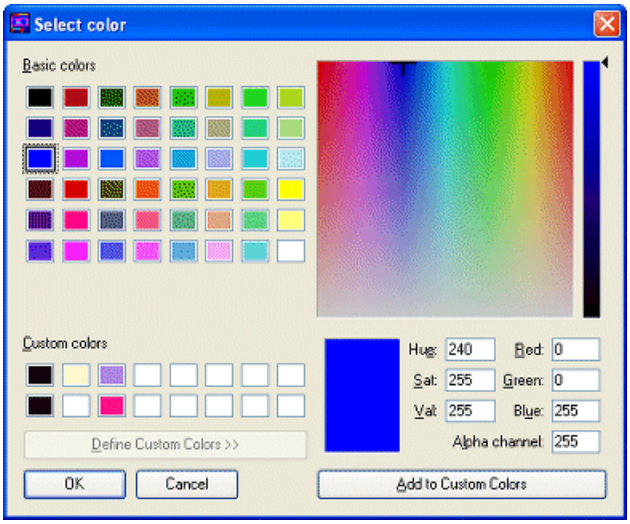


Figure 11-9: Select Color Panel

Color Panel

Color panel is used both for creation and selection of a layer’s color.

Layer Color Selection

This color setup panel, invoked from both **Layer setup** and from **Application color setup**, allows you to select color for a selected layer in two ways:

- by directly choosing a predefined color in the basic color field.
- by defining a custom color with following steps:
  - select by mouse click an empty box in the Custom colors field,
  - define the custom color by entering numeric HSL (hue-saturation-luminance), or RGB (red-green-blue) values, or by clicking in the color palette (the **Alpha Channel** parameter does not matter),
  - click on the **Add to Custom Colors** button and press **OK**.

The syntax of the technology file specifies color RGB value. For example:

```
Color = (0,255,255)
```

Background Color

The **Setup→Editor/Viewer...→Viewing** form allows you to select either black or white background color. In the case of a white background, the default color of all auxiliary markers will be inverted from white to black. If custom colors are used for these auxiliary drawings, you must assign them for white and black backgrounds separately using **Setup→Hotkeys/Menus/Colors...** command.

Stipple Setup

Stipple patterns for layers are selectable from stipple libraries.

After a stipple pattern is selected for a layer, it is stored in project's technology and saved in technology files (**Setup→Technology→Save Technology File...**).

Stipple Pattern Libraries

EXPERT allows the use of two kinds of stipples for the layer display:

- The first type belong to built-in stipples libraries (system library which contains cross, lines, and dotted stipples of different densities, and X-stipples library which contains several kinds of complete X or partial X stipples),
- The second type belongs to the external user-definable library of stipple patterns. To invoke this feature, use the new **Edit** button on the **Stipple Selection** panel (command: **Setup→Technology→Layer/Wire Setup...→Stipples...→Edit Bitmaps**). The stipple editor panel will be displayed. It allows defining and managing of user-defined libraries (see Stipple Editor Operations below).

In any case, a stipple pattern can be assigned to a layer in this way:

- In the **Layer/Wire Setup** panel, select the layer you want to assign a stipple.
- Click on the **Stipples** button: the **Stipple Selection** panel appears (see Figure 11-10).
- In the Stipples tree view, expand the item of the library containing the desired stipple (the external library is under the "Bitmaps" item).
- Select the stipple and click **OK**.

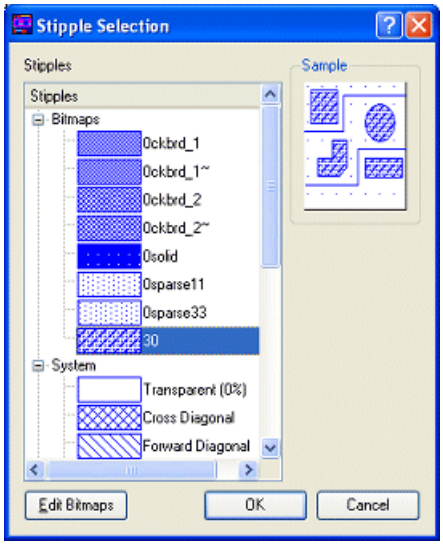


Figure 11-10: Stipple Selection Panel

Note: External stipple libraries files have a `*.stp` extension. The default filename, which resides in EXPERT'S installation directory, is default.stp.

Although stipple libraries are ASCII text files, editing them manually is not recommended, except for renaming the stipples and moving around blocks of stipple definitions as a whole.

Stipple Editor Operations

The **Stipple Editor** dialog panel consists of the following controls. In the **Stipple manipulations** part, you can modify or create a stipple:

- The **Preview** window shows layout samples for the current stipple.
- The **Bitmap** window shows enlarged structure of bitmap as a grid of bit cells. You can edit this structure by mouse clicks. The color of a cell can be changed by a mouse click. **Preview** window is updating immediately after any modification of bitmap structure.
- The **Rows and Columns** drop-down menus change the number of bitmap rows or columns; allowed values are between 1 and 32.
- The **Clear** button sets to black all bits of the **Bitmap** window.
- The **Invert** button inverts colors of current bitmap.
- The **Accept** button stores modifications of the current bitmap.
- The **Reject** button discards any changes of the current bitmap made since the last "Accept" operation.

The **List manipulation** part contains names and sample images of all stipples of the current library. A blue rectangle around a stipple name marks the current stipple image, which is shown in the **Preview** and **Bitmap** windows. You can change the current stipple by key arrows, scrolling the bar, or mouse clicks.

- The **Copy** button adds to the list of available bitmaps a copy of the current bitmap. You must enter the name of the newly created bitmap.
- The **New** button adds to the list of available bitmaps an empty bitmap. You must enter the name of the newly created bitmap.

- The **Delete** button removes the current bitmap out of the list of available bitmaps. You are asked to confirm the operation.
- The **Import** button adds to the list of available bitmaps, a copy of the bitmap contained at another library of stipples. This button invokes the **File Selection** dialog box (to point out the library file) and a stipple selection dialog (to select a stipple defined at this library).
- The **Merge** button adds to the list of available bitmaps all bitmaps contained in another stipple library. You are prompted to point out what library file is to be merged.

The **File operations** window contains the name of the active stipple library.

- The **Clear** button creates an empty list of available bitmaps. It helps you to create a new stipple library.
- The **Load** button invokes the file selection dialog box to point out the new stipple library to be activated instead of the current library.
- The **Save as** button invokes a file creation dialog box which allows you to save modifications made in the current library, as well as save them into a new library file.
- The **Apply to technology** button instantaneously spreads to the **Layer/wire setup** panel a change of stipple, which has been done in the **Stipple Editor**.

Usage in Project

The **Stipple Editor** dialog panel can be closed by hitting the `Esc` key, or by clicking the **Close** button. If you did load a new stipple library (or rename the current library via the **Save as** operation), you are prompted to confirm that the last loaded library will be used with the project in the future.

The last library name used is saved in the configuration file (`*.ecf`), so in fact there are two ways to change the stipple library:

- from **Setup→Technology→Layer/Wire Setup...→Stipples...→Edit** command or;
- by loading a custom `.ecf` file.

After closing the dialog panel, you are returned to the stipple selection scenario.

Wire Setup

Every layer may have its own default settings for wires created in it. The dialog shown in Figure 11-11 allows you to input the necessary parameters, as described in Chapter 2: "Basic Notions", "Primitives" on page 2-4. It is invoked by clicking the **Wire Style** button of the Layer setup.

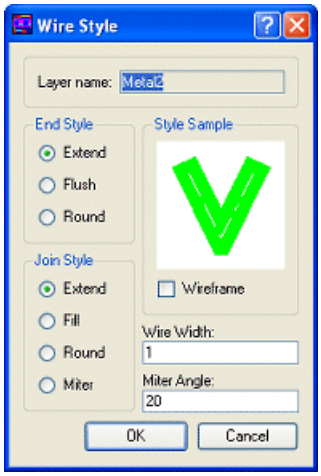


Figure 11-11: Wire Style Setup Dialog

Zero-Width Wires

Sometimes it is necessary to make marks in the layout (e.g., mark centerlines, alignment, etc.) by lines, in other words, by one-dimensional objects. In EXPERT, it is recommended that wires of zero width be used.

Boxes and polygons of zero thickness are not allowed. Even if they are imported from a GDSII file, they will be purged either during editing or saving operations.

Scratch Layers

The “Scratch layer” flag is added to layer’s properties in **Setup→Technology→Layer/Wire Setup...** to mark non-drawing layers; those intended to hold temporary data. The corresponding parameter in the technology file is:

```
ScratchLayer = TRUE / FALSE.
```

Note: In the **Layer Bar**, if the cursor hovers over a scratch layer, the “Tool Tip” text that describes the layer contains an “s”.

Edge Layers

The “Edge layers” flag is included in the **Layer/Wire setup** panel to mark layers containing line segments generated by DRC operations from boundaries of shapes on shape layers (that contain polygonal shapes only). Therefore, an edge on an edge layer has 'inside' and 'outside' notions properly defined. In a technology file, the corresponding parameter is:

```
EdgeLayer = TRUE/FALSE.
```

Note: In the **Layer Bar**, if the cursor hovers over an edge layer, the "Tool Tip" text that describes the layer contains an “e”.

Derived Layers

A **Derived Layer** is a layer whose contents are produced by certain operations involving other layers. In the current version there are three types of layer operations: logical, resizing, and selection. Layer Derivation is the (process of) definition of a layer in terms of other layers. It may be performed either via **Layer Setup** dialog or by writing the corresponding commands into the technology file. Layer Generation (or Layer Rebuilding) is the process of actual construction of the contents of derived layers according to their definition. It may be invoked in several ways:

- from the menu: **Tools→Derived Layers**
- on activation of a derived layer using the **Layer Bar**
- on switching to a window where a derived layer is the active one.

Restrictions

When a single layer is being rebuilt, the layers from which it was derived may be non-rebuilt. Therefore, it is recommended that you use the menu command, **Tools→Derived Layers→Generate All Derived Layers,** which regenerates derived layers in their logical order.

See also Chapter 12: “Tools”, Section 12.14: “Handling Derived Layers” for more information.

Derived Layer Definition Using Layer Setup

The **Layer Setup** panel provides tools for maintaining derived (generated) layers in GUARDIAN DRC. They are the **Derived layer** check box and the **Derivation...** button.

The Derived layer check box has a dual function:

- On one hand, it indicates whether the current selected layer in the layer list is a derived one (exactly like the boxes of the Derived column in the Layer list of the **Layer Setup** panel).
- On the other hand, it allows you to control the type of the current layer that turns ordinary layers into derived layers and vice versa. If the Derived layer box is not checked (left blank), it means that the layer currently selected in the layer list is not a derived layer. By checking this box, you are declaring this layer to be a derived layer.

Note that there is some danger in this action: When setup is complete, all current contents (if any) of the layer will be lost at the first generation of derived layers. Therefore, you are asked first to confirm your intention to set the selected layer to be a derived one. After confirmation, the **Derivation...** button becomes active, allowing you to specify the required derivation parameters (see below).

If the **Derived layer** box is checked, it indicates that the current layer in the layer list is a derived layer. Unchecking this box to stop derivation of the layer and turns it into an ordinary layer. All current contents of that layer (the result of the last layer generation) remain unchanged and may be edited in the regular way after layer setup is complete. You should confirm your intention to stop layer derivation because it results in the loss of all its derivation parameters, including the derivation macro (see Section "Layer Derivation Editor" for details).

The **Derivation...** button invokes the **Layer Derivation Editor** dialog from the **Layer Setup** dialog, allowing you to control existing properties of derived layers (if any) without defining new ones. If this button is pressed while the current layer in the layer list is derived, the **Layer Derivation Editor** dialog is displayed on the screen with derivation parameters of that selected layer.

Layer Derivation Editor

Layer Derivation Editor (Figure 11-12) provides control over all the existing derived layers. It allows you to create, inspect or modify the macro defining any derived layer.

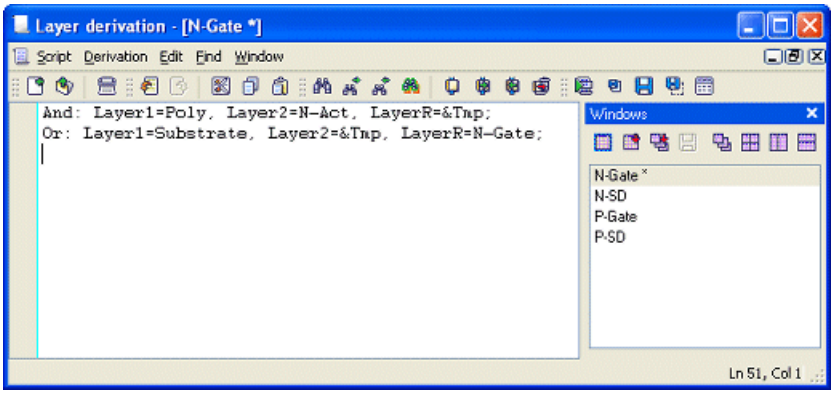


Figure 11-12: Layer Derivation Editor

Here is the list of available operations on layers to create a derived layer macro:

- **Logic operations:** AND, OF, DIF, XOR and MERGE,
- **Select operations:** Inside, Outside, Hole, Connect, Cut, Overlap, Enclose and Touch,
- **Resize operations:** Oversize, Undersize and Size,
- **Check operations:** Width, Outdistance, Indistance, Intersection, Length…
- **Layer operations:** Substrate generation, Copy and Delete.

These commands can be written directly in the **Layer Derivation Editor** or they can be generated thanks to the items of the Insert menu.

You may find the exhaustive description of the syntax of these five types of commands in the DRC part of GUARDIAN manual, respectively in Sections 2.3.5, 2.3.4, 2.3.7 and 2.3.8. The use of the first four panels of the **Insert** menu is described in Section 2.6 of GUARDIAN manual ("Interactive operations"). The **Layer Operation** panel is described in Section 2.4.2.3 of GUARDIAN manual ("Layer Operation Dialog").

When you define the derivation macro of several layers, use the commands of the **Derivation** menu of the **Layer Derivation Editor**. These commands, which help to check syntax and coherence of macros and allow temporary changes, are described below. (All other generic menus of the **Layer Derivation Editor** are described in the SEDIT user's manual.)

Note: When the **Layer Derivation Editor** is opened, the **Layer Setup** dialog can be still accessed. It allows you to open several scripts of layer derivation thanks to the **Derivation** button of the **Layer Setup** panel. It is then possible to spread any modifications on these opened scripts thanks to the Accept (all) derivation commands which are described below.

Syntax checking: The **Layer Derivation Editor** allows checking the syntax of the macros defining derived layers.

This function can be accessed through the menu **Derivation→Syntax**. It will check the current macro in regards to the syntax and the coherence of the layer names, and to available names in the technology.

Save the macros: In order to save the defined macro in the technology, the menu **Derivation→Save to layer setup** must be used.

Show full derivation: This feature allows you to build and watch whole derivation scripts for a particular layer. Usually you describe a derived layer by writing several commands, but these commands can contain other derived layer(s) and so on. If you start "show full derivation" you receive derivation script for the current layer with all intermediate commands/layers beginning from input (non-derived) layers.

Accept derivation: This command means that all changes in derivation of current layer will be checked for right syntax and derivation order and then temporarily stored as derivation of this layer.

If you made any changes in the layer derivation and did not use this command, syntax checking for another layer will use the derivation of the previous layer without its changes.

Accept all derivations: If several scripts of layer derivation have been opened, this command applies "accept derivation" for all currently editing layers.

Note: If you used "Accept derivation" or "Accept all derivations" commands but did not use "Save to layer setup", all changes will be discharge after you leave derivation setup menu.

Multiple Operations Macros

One single derived layer can be created by a sequence of several operations. The rule, which must be respected in such a macro, is to create only one derived layer per macro.

Here is an example of multiple operations macro:

```
AND: Layer1=Poly, Layer2=N-Act, LayerR=Tmp;
OR: Layer1=Substrate, Layer2=Tmp, LayerR=N-Gate;
```

In this sequence of two boolean operations, the final generated derived layer is `N-Gate`. It is defined by logical AND between layers `Poly` and `N-Act`, and then by logical OR between layer `Substrate` and resulting layer of the first operation `Tmp`. This layer is considered as a derived layer too, so this macro

does not obey the above rule (create only one derived layer per macro). There are three solutions to this problem:

- You can split this macro into two individual ones. The drawback of this method is the increase the size of the techno file, and the number of layers displayed in the **Layer** bar.
- You can define the intermediary layer `Tmp` as invisible to the technology. This is done by adding "&" character at the beginning of its name. This leads to the following macro:

```
AND: Layer1=Poly, Layer2=N-Act, LayerR=&Tmp;
OR: Layer1=Substrate, Layer2=&Tmp, LayerR=N-Gate;
```

- You can use the "Logicform" command which allows defining several logical operations in one single command (see detailed syntax in Section 2.3.8.2 of the GUARDIAN manual). In this example, the macro would be:

```
LogicForm: N-Gate=Substrate.or.(Poly.and.N-Act);
```

Derived Layer Generation

When all the derived layers are defined, you can generate them through the menu **Tools→Derived layers**. The detailed description of the available commands of derived layers management is in Chap.12. Section 12.12: Handling derived layers)

Note: The derived layers are all generated in with a "Flat" scope in the current cell. It means that EXPERT makes a flat copy of the current cell and uses it as input instead of the current cell. All resulting derived layer objects are placed in the active cell.

Derived (Generated) Layers

Layer specification within the technology file allows definitions of derived layers. A derived layer is obtained from other layers (input layers) by execution of a given sequence of logical, selection and resizing operations. The following example shows layer description if the layer is the derived one.

```
Layer
  {
  Name = "P-Gate"
  Wire
    {
    MiterAngle = 20
    Width = 1
    Joint = EXTEND
    End = EXTEND
    }
  Stipple = "0solid"
  Bitmap = "0000000008000000080000000200000001000100FF00FF00FF00FF00FF0"
  Color = (0,255,0)
  ColorName = "green"
  GDS2Num = 17
  GDS2DataType = 0
  Derive = "And: Layer1=Poly, Layer2=P-Act, LayerR=P-Gate(17);"
  Scope = CELLWISE
  UseGridParams = FALSE
  ScratchLayer = FALSE
  EdgeLayer = FALSE
  }
```

11.4.4: Connectivity Setup

Connectivity information is specified in the technology file as sets of layers connected by a connecting layer, or a via layer. The format is given by examples:

```
connect = ("layer1" "vialayer")
connect=("layer1" "lay2" "lay3" "vialayer")
connect=("layer1", "lay2", "lay3";;;"vialayer")
```

The connectivity list may consist of any number of layer names.

Layer names must be in quote marks [“ ”].

The last layer is the connecting layer for the remaining ones.

`connect=(  )` are obligatory elements of syntax.

You may use any separators between layer names for readability.

Connectivity Setup Dialog

This dialog is invoked by **Setup→Technology→Layer Connection Setup...** menu command. It has three columns. The first column is for selection of a layer whose connectivity must be changed.

This column has two radiobuttons: **Connecting Layer** and **Layers In Connections**.

“Connectivity Layer” Radiobutton is Selected

In this case, the layer selected from List1 is considered to be the connecting via (layer). List2 contains the layers connected through the given via. List3 contains the layers not connected by the given one. You may move layers between List2 and List3 by using the arrows located between them.

As an advanced option, one layer may define several connectivity sets. The combo box below List1 selects the number of the required connectivity set defined for the given connecting layer. You must first click on a layer in List1, then select the set number from the combo box.

If only one set is defined for a connecting layer, simply double-click on it in List1.

Attach label only: This option should be checked if you want that only labels of layers in List2 are taken into account in the connectivity of layout objects. In this case, the shapes of layers of List2 will not be connected to the via layer.

“Layers In Connection” Radiobutton is Selected

In this case, for the layer selected in List1 you may assign the connectivity sets to which it must belong.

List2 contains all connectivity sets to which the selected layer belongs.

List3 contains the remaining connectivity sets.

You may add a layer to a set by moving this set from List3 to List2 and vice versa.

11.4.5: Device Setup

Device information is specified in the technology file in order to allow netlist extraction tool (GUARDIAN LPE) to extract all devices which are designed on the layout.

The description of this panel and the list of all kind of devices which can be extracted by GUARDIAN LPE can be found in GUARDIAN manual, section 3.4.3 “Devices”.

11.4.6: Grid Setup

There are several kinds of grids in EXPERT. Their sizes and visibility are defined in **Grid Setup**. The grid setup dialog described below is invoked from three different places, for different purposes.

1. **Default grid settings** are set from **Setup→Technology→Grid Setup...** or directly with the **Grid Setup** button of the **Command** toolbar. These grid settings applied to all editing operations, if they are not overridden.
2. **Individual layer grid settings** are set from the **Layer Setup** panel (**Setup→Technology→Layer/Wire Setup...** menu command). They are applied to that particular layer.
3. **Cell instance placement grid** (see Section 11.4.9: "Instance Grid") is set as a parameter of the cell instancing dialog and is applied to cell instance/array placement. See Chapter 5: "Creating Layout Objects", Section 5.3.1: "Instance/Array Attributes" for more information.

Grid Setup Panel

Steps for snap, visual and super grids may be set differently in X than in Y. This is controlled by the option, `"Equal grid steps in X and Y"`, in the **Grid Setup** menu.

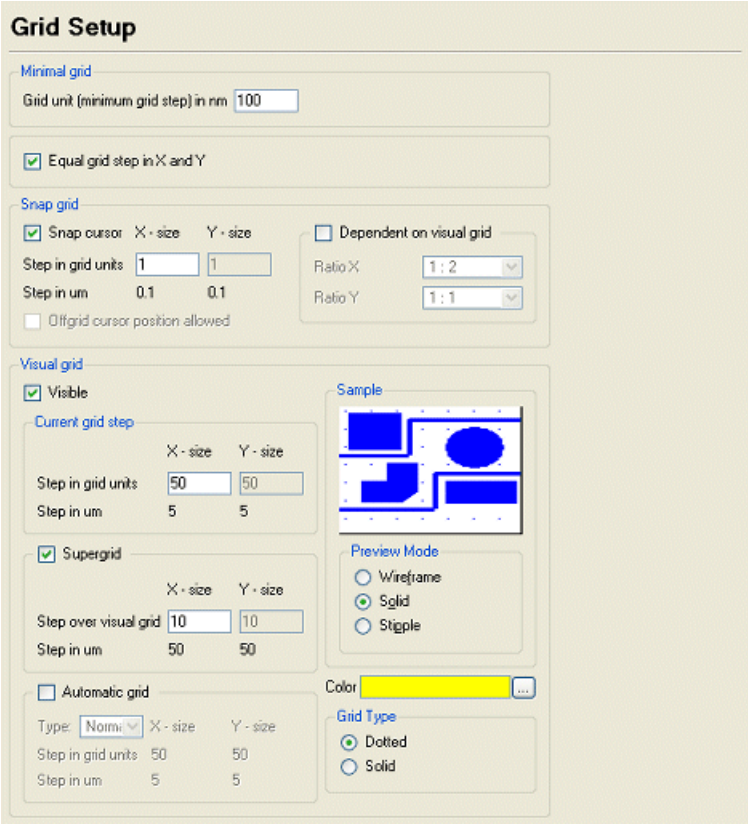


Figure 11-13: Grid Setup

Minimal Grid (Grid Unit)

It is the grid with the minimal step allowable under the current technology. This minimal step is called **grid unit** in EXPERT. This value is also called **resolution** or **spot size** in other CAD systems. This step is stored in the technology file as a positive integer "Grid" value, in nanometers. This step is the minimal step allowable in the "Snap cursor" mode.

Snap Grid

This is a grid that controls cursor movement: cursor stops only at snap grid nodes. Snap grid step is a positive integer in terms of minimal grid step. For example, if grid unit is 10 nm and snap grid setting is 20, then the actual value of the snap step is 10x20=200nm=0.2u.

When the snap cursor box is checked (or if the equivalent command **Edit→Snap cursor to Grid** is activated), the cursor is snapped:

- by default to the grid which is defined in the **Step in grid units** field of the **snap grid** part of the **Grid** panel,
- to a ratio of the visual grid if the **Dependent on visual grid** box is also checked. In this case, the attached fields Ratio-X and Ratio-Y selecting the ratio between **Snap grid step** and **Visual grid step** must be filled. The grid step is maintained while changing the visual grid step anyway.

When the **Snap cursor** box is not checked, the cursor is off-grid in all non-design modes (void tool, select tool, ruler...). This is particularly useful when performing measurements. But when activating functions of shapes design or shapes modification, the cursor is automatically snapped to the **Snap grid step** or to the **Visual grid step**, which are defined above (depending on the **Dependent on visual grid** box).

Visual Grid

The **Visual grid** is the one shown on the screen (and on plot). The Visual grid step is also given in terms of minimal grid step. A Visual grid may be either an **automatic** or a **fixed-step**. In automatic mode, the grid most naturally subdivides the screen (however, its step cannot be less than that of the minimal grid). In fixed-step mode, the grid step is always the same, specified in **Grid Setup** as a multiple of the minimal grid step. Therefore, this grid is invisible when it is either too small or too large with respect to the screen.

Supergrid

Supergrid nodes are marked by small crosses. The **Supergrid** step is given in terms of the visual grid step. It may be switched on/off.

Grid Characteristics

The grid may have the following characteristics.

- The **Dotted** grid is represented by dots at the grid nodes.
- For a **Solid** grid all grid lines are drawn.
- **Supergrid** and **Visual grid** may be set to be either visible or invisible.
- Grid step may be set to **automatic** or manual. In the latter case you can input the grid step size, in current units.
- If the grid is **automatic**, you can select its density by choosing **Automatic grid type** to be one of: normal, dense, or sparse.
- You may select the grid's color by clicking the **Color** button. The same color palette tool as for layer setup is used.
- Snap, Visual, and Supergrid steps in X and Y directions may be set separately.
- Any change in the **Grid Setup** can be previewed in the **Preview Mode** area of the **Grid Setup** dialog.

11.4.7: Pwire Templates

This page is to add, remove, or modify Pwire templates in the technology section of the EXPERT database (see Figure 11-14). The Pwire template is stored in the technology with a unique name, and used for quick creation of Pwire objects (see Section 5.6: "Parametric Wire").

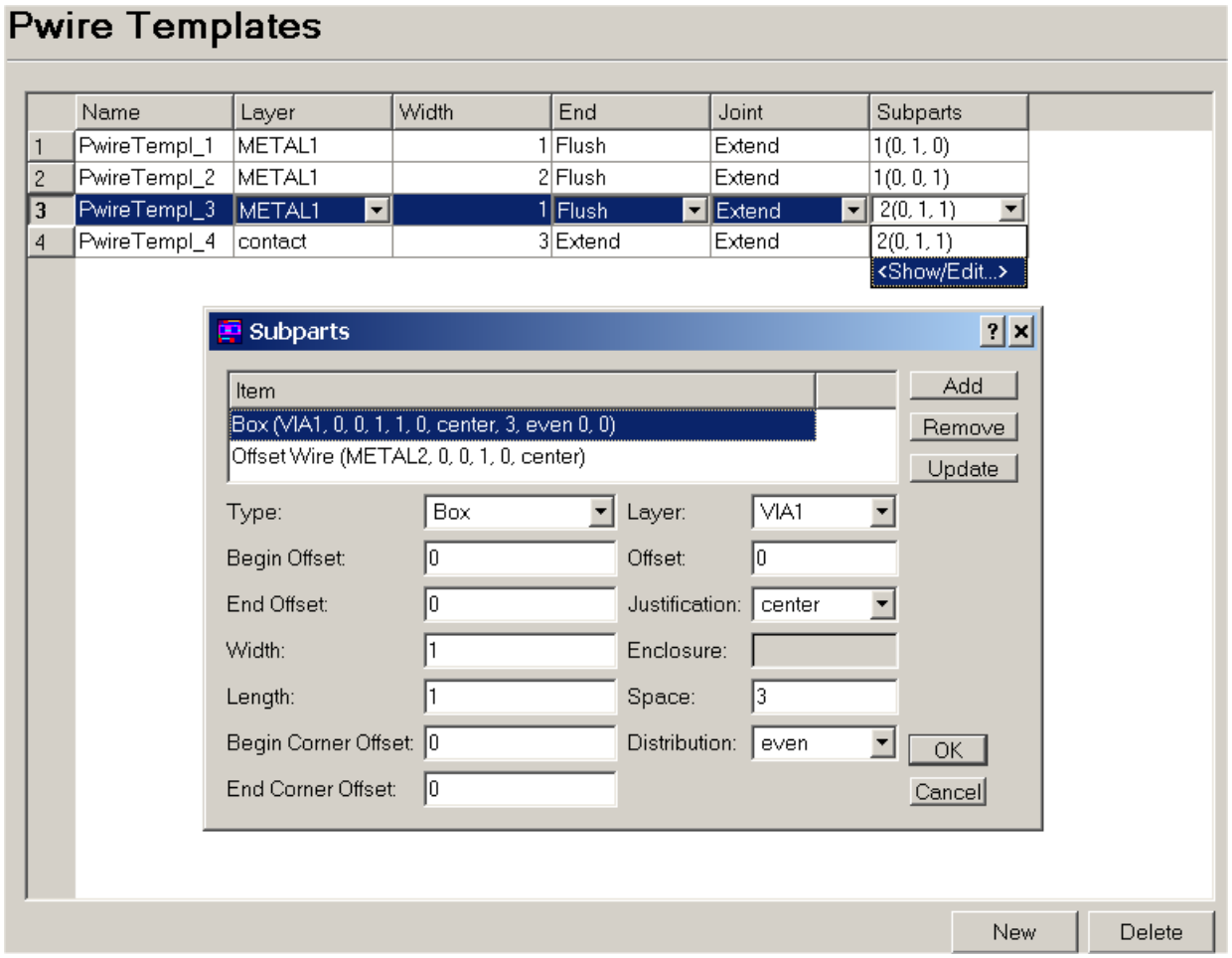


Figure 11-14: Pwire Templates Setup Page

Use the **New** button to add a new template, and **Delete** to remove a selected template from the technology. A detailed description of each Pwire template parameter, and all subparts parameters, is provided in Section 5.6: "Parametric Wire". Removing Pwire templates from the technology does not delete Pwires from the layouts.

To add a Pwire template to the technology, you can also use the Xi command `PWIRE_TEMPLATE`, or export a Pwire template definition from the EXPERT external `ASCII *.tcn` technology file.

11.4.8: Contact Setup

This page is to add, remove, or modify the contact templates in the technology section of the EXPERT database (see Figure 11-15).

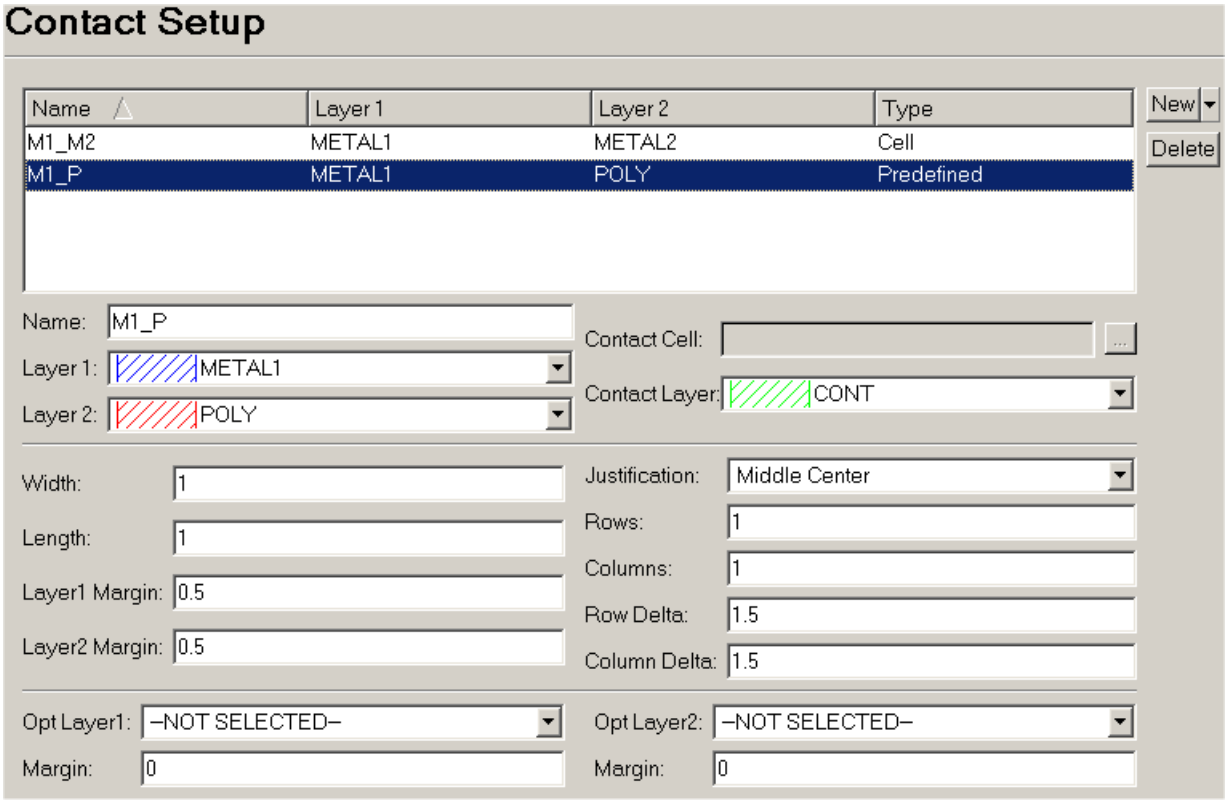


Figure 11-15: Contact Setup Page

A contact template is an EXPERT contact with fixed basic parameters stored in the technology with a unique name. A contact is an EXPERT predefined device for connecting two layers, and has two sets of parameters: basic and user. The basic parameters define the physical content of the contact: contacted and optional layers, contact layer. The user parameters (contact width, length, etc.) can be modified when a contact is placed into a layout. The default values for the user parameters are also stored in the technology. The contact pattern is depicted in Figure 11-16.

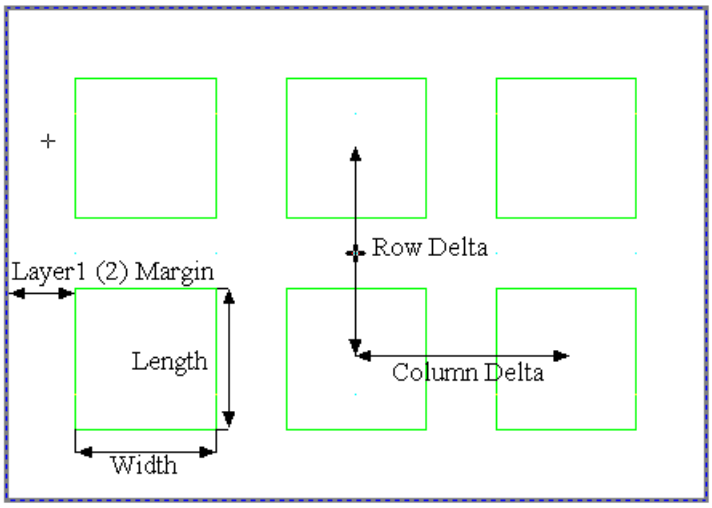


Figure 11-16: Contact Pattern

The following parameters set the contact template.

Basic parameters:

- **Name** sets the contact name, used when instance of contact is placed;
- **Layer 1** and **Layer 2** specify two contacted layers;
- **Contact Layer** specifies layer for contact boxes;
- **Layer 1** and **2 Margin** sets overlap value for Layer 1 and Layer 2 respectively (see Figure 11-15);
- **Opt Layer 1** and **2** specifies optional layers to be used with contact definition;
- **Opt Layer 1** and **2 Margin** specifies overlap values for corresponding optional layers.

User parameters:

- **Width** and **Length** set the width and length of the contact;
- **Rows** and **Columns** set the number of rows and the number of columns in a contact array.
- **Row Delta** sets the distance between the centers of the contact rows (see Figure 11-16);
- **Column Delta** sets the distance between the centers of contact columns (see Figure 11-16);
- **Justification** sets the origin of a contact array.

Note: If basic parameters are modified, then all instances of the contact are recalculated. If default values of user's parameters are modified, then each default parameter in all instances is updated and recalculated if needed.

11.4.9: Instance Grid

When instantiating a cell or an array of cell in Expert (with command **Edit→Create Object→Create Instance/Array**), the cell instance positioning is performed according to the default snap grid set by the **Grid Setup** panel. If you want to have an instance position according to a different snap grid, you should activate the menu **Setup→Technology→Instance Grid Setup...**. You will see the standard grid setup panel, described in Section "Grid Setup Panel". In this window you can change the grid characteristics so that they will be activated when placing a cell on the layout.

11.5: GDSII I/O Setup

Setup→GDSII Input/Output... or **Setup→Setup...→GDSII Setup→GDSII Input/GDSII Output** provide the following settings:

On GDSII Input

- **GDSII file extension(s):** Filters file lists in file browser.
- **Ignore layer undefined in technology:** Skips layers not defined in the technology file.
- **Ignore DATATYPE:** Sets DATATYPE value to 0, effectively merging layers with the same GDSII number but with different datatype values.
- Ignore magnification for cell instances/arrays.
- Use database unit from GDSII file.
- Convert rounded or extended 1-point wires.
- Skip GDSII BOX data records.
- Write box-shaped BOUNDARY records as EXPERT boxes.
- Convert Text Presentation from Older EXPERT.
- **Forced Line End and Join style:** Overrides PATHTYPE defined in the GDSII file for all layers.
- **Self-intersection check for regions and wires:** Copy or move self-intersecting polygons to error cells.
- **Non-orthogonal arrays:** Copy or move non-orthogonal arrays to error cells.
- **Write GDSII input log file:** GDSII input log file contains enhanced individual cell and layer object statistics.
- Layer remapping.
- **Use text rescaling option in GDSII Input/Output:** Resizes text to match the requirements of a foreign tool.

On GDSII Output

- Output GDSII file extension for GDSII files.
- Cell name limit: max number of symbols.
- Convert cell names into upper case: matches GDSII strict standards.
- Convert cell names to lower case.
- Representation of arrays on output.
- Convert wires to regions.
- Options for writing cells from external libraries.
- Write derived layers: saves derived layers into GDSII file.
- Write scratch layers: saves scratch layers into GDSII file.
- Use user defined DB units.
- Use internal DB units.
- **Cut large polygons/wires (default):** Instructs you to cut polygons/wires into smaller parts to match GDSII strict standard restriction on vertex count.
- Restrict layer numbers and datatypes.
- **Write GDSII output log file** - GDSII output log file contains individual cell and layer object statistics.
- Use output layer remapping table.

The GDSII Output Setup dialog is shown in Figure 11-17.

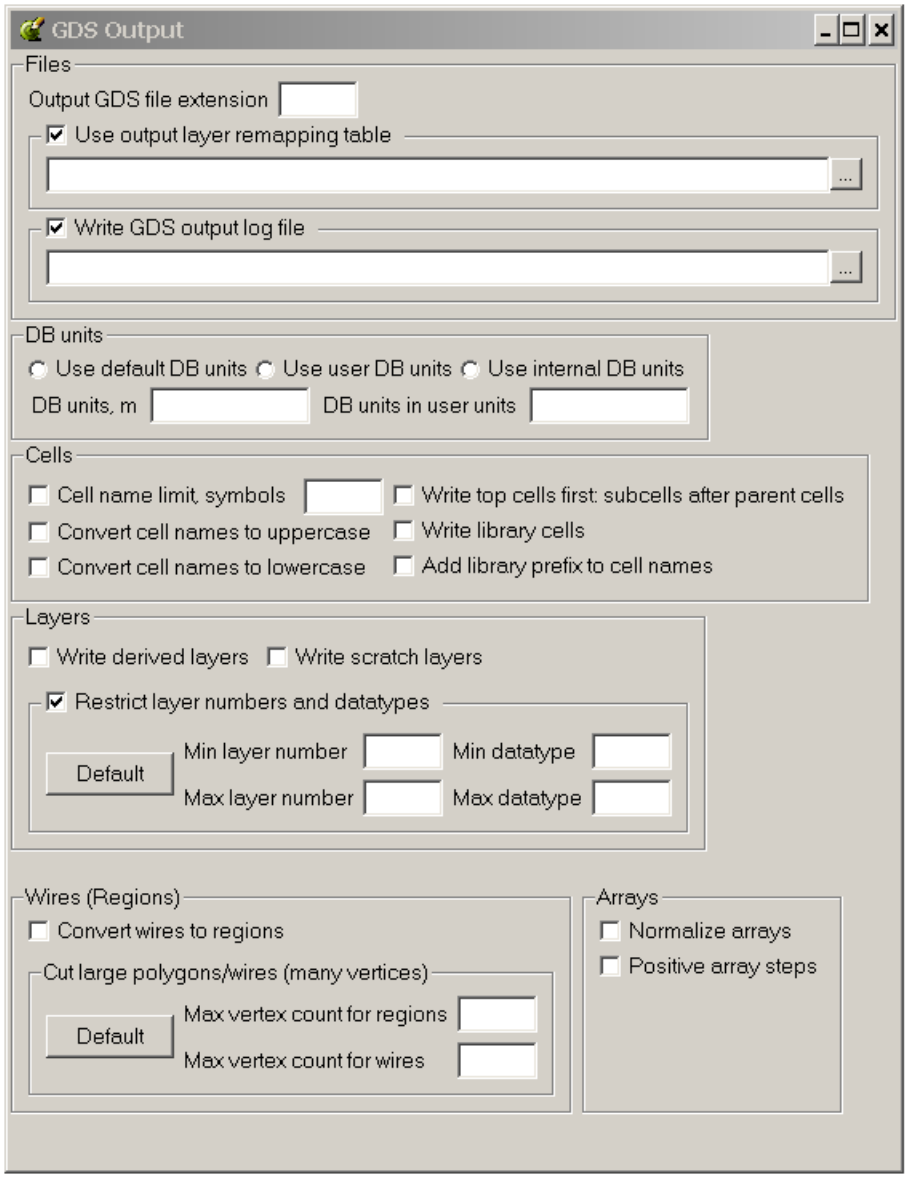


Figure 11-17: GDSII Output Setup Dialog

11.5.1: Description of Particular Options

GDSII Input Panel

GDSII file extension(s)

The **GDSII file extension** field in the **Input** group inputs the list of possible extensions of GDSII files (e.g., like this: `*.gds; *.sf` (a list of file specifications are separated by a semicolon) or asterisks *.

This list is used in the **File→Open...** and **File→Import...** dialogs to filter the file list in the **File Browser**.

Ignore layers undefined in technology

GDSII Input setup page has the check box **Ignore layers undefined in technology**. If this option is on, only layers defined in the selected technology file will be read from the GDSII file. Otherwise, all layers with GDSII numbers and DataTypes in the 0-255 range will be read.

Ignore DATATYPE

Use technology files to assign layer names to GDSII layers, as well as the **Ignore DATATYPE** option. If the setting **Ignore DATATYPE** at the setup **Setup→GDS Input/Output...→GDS Input** panel is on, then the DataType value for all geometry layers will be set to 0.

Note: The TEXTTYPE field of GDSII file is mapped into DataType value of EXPERT'S layers. Therefore, you may see some layers with non-zero DataType, even if the **Ignore DATATYPE** setting is on. These layers contain only text.

Ignore Magnification for Instances/Arrays

This option is added for compatibility with some other ECAD systems that ignore the MAG record of GDSII format for cell instances/arrays, and may write undefined values into this record.

Use Database Units from GDSII File

If the **Use Database Units form GDSII File** option of the GDSII input setup (menu **Setup→GDS Input/Output...→GDS Input**) is on, then EXPERT'S Database Unit value is taken from the GDSII file (**database units** part of GDSII UNITS record), if it is one of 0.0001um, 0.001um, 0.01um, or 0.1um. Otherwise the current EXPERT setting is applied.

This setting is recommended when a wafer-scale layout is loaded from a GDSII file.

In the case when the option is not checked, the input GDSII file will be converted to the DB unit value, which is specified in the menu **Setup→Editor/Viewer...→General** in the field **Database unit for new/imported projects**. See Section 11.3.1: “General Page” for more details

Convert Text Presentation from Older Expert

In very old versions of EXPERT, the default alignment of the text, with respect to its origin (reference point) was the lower left corner, while the default of GDSII format is the upper-left corner.

Now it is possible to select text alignment in the numeric input for text creation/modification. There are three vertical and three horizontal alignments; nine choices total.

If you need to read a GDSII file written by an old version of EXPERT, use the setting “Convert text presentation the previous version of EXPERT" found in the **Setup→GDS Input/Output Setup→GDS Input** menu.

Import Layer Remapping

Import layer remapping table works in a similar way as export layer remapping table (see below).

GDSII Input/Output Panel

Text Rescaling

If the **Use text rescaling** option is OFF, text size in microns equals Magnification Factor value (GDSII record type 27 MAG). If omitted, a magnification of 1 is assumed.

Text rescaling value allows text size adjustment when transferring GDSII files between different CAD systems. For example, if the texts appear too large after loading a GDSII file, try to decrease this value and reload the file.

By default **Use text rescaling** is ON for compatibility with previous versions of EXPERT.

Restrict property attribute number

You can set up a specific limit for property attribute numbers to be written into the GDSII output stream. It can be done by checking the **Restrict property attribute number** box and specifying the max number of property attribute allowed to write into output stream. The default range for property attribute numbers is an integer from 1 to 127. Properties with invalid numbers will be skipped. A warning will be displayed if properties with attribute numbers exceeding the specified limits are encountered during GDSII saving.

If the **Restrict ...** option is not checked, all properties (both user-defined and EXPERT internal) will be streamed out into the output file. It is important to uncheck this option if you want to preserve the connectivity information stored in the EXPERT internal properties.

GDSII Output Panel

Cell name limit

To comply with GDSII standards limiting the cell names to 32 characters, you can set a maximum limit to cell name character number.

Output Representation of Arrays

Array deltas (steps in x and y directions) may be negative for the untransformed in EXPERT. While this is not forbidden by the GDSII format specification, some systems do not accept such values or interpret them incorrectly.

If the **Normalize arrays** option is ON, then such arrays are written into a GDSII file in a modified way. The actual geometry remains unchanged, but the first array element is considered to be in the lower left corner of the whole array and deltas become positive.

A similar modification is applied to arrays with transformation: rotation and/or mirroring. The first array element is considered to be in the corner into which the lower left corner of the array is moved if the transformation is applied to the array considered as "solid body".

An alternative option is **Positive Array Deltas**. If this option is checked, then the deltas for the arrays written into a GDSII file are always positive, irrespective of array transformation.

Writing External Library Cells into GDSII File

If a project has instances of cells from other libraries, the following two options define how their external cells (and their instances) will be written into a GDSII file:

Write library cells

With this option on, cells from external libraries referenced in the current project will be written in the output GDSII file as well.

Add library prefix to cell names

With this option on, cells from external libraries will be renamed by prefixing it with the logical library name. For example, cell `nand2` from library "gates" will be renamed `gates_nand2`. Its instances will be changed in the same way.

This option is useful if the project has cells with the same names from different libraries.

Loading GDSII files with Missing Cells (Unresolved Cell References)

To restore library references after saving a project file with the **Write library cells** option OFF, you must do the following:

- Read the saved GDSII file as usual. A message appears warning you that there are unresolved cell references.
- Go to into the **Libraries→Setup...** menu command and activate the libraries with the required cells. If necessary, add the required libraries to the setup.

Note: Cell references will be resolved in the same order the libraries were activated. If a cell is unresolved, and a cell with the same name exists in two libraries to be activated, then the cell will be taken from the library which was activated first.

Database Units on Export to GDSII format

Three variants of settings are possible:

- **Use default DB Units**
- **Use user DB Units**
- **Use internal DB Units**

If the **Use default DB units** button is checked, the database unit (DB unit) of the output GDSII file will be set to 0.001um automatically (it corresponds to Cadence default value).

If the **Use user DB Units** button is checked, settings of GDSII Output setup allows you to change the value of the UNITS record of GDSII file format (database unit and user unit) on export to GDSII. Then it is necessary to fill the two activated fields DB units and DB units in user units.

With the option **Use internal DB units**, current internal DB units (0.0001um, 0.001um, 0.01um or 0.1um) will be used. (The current internal DB units can be checked in menu **Setup→Editor/Viewer Setup→General**).

On import, the database unit and user unit values are stored in EXPERT'S project file and are used for exporting this project back into GDSII format.

Restrict Layer Numbers and Datatypes

It consists in the following fields:

- **Restrict layer numbers and Datatypes** box
- min/max layer number
- min/max datatype value
- default values (min=0 / max=255)

If the **Restrict** option is checked, it is possible to input the maximum values of the layer number and the datatype allowable for output. No layer with these parameters exceeding the assigned limits will be written into the output files. When saving under GDSII format a project containing layers exceeding the limits, a warning will be displayed with the list of involved layers.

If the **Restrict** option is not checked, all layers will be written into the GDS file.

Export Layer Remapping

Layer remapping tables provides the following capabilities:

- To export projects or cells into GDSII format using layer names and datatypes which are different from those used in the technology file.
- To save into GDSII file only selected layers.
- To merge several EXPERT layers into one GDSII layer on export.

The Export layer remapping table is a file consisting of lines in one of the following formats:

```
<LayerName> <GDSII_Number> <DATATYPE>
```

or

```
<LayerName> <GDSII_Number>
```

In the second case, DATATYPE is set to 0.

Examples

```
Via     10 0 Poly     6 GRings  43 5
```

To apply remapping, use the following settings in the **Setup→GDSII Input/ Output→Output** panel:

- Check the **Use layer remapping table** option.
- Select the file with the table, either by typing its name in the "File name" field, or by browsing the files by clicking the **Browse** button (labeled by "**...**").

If you are using the remapping table, only those layers that are listed in the table will be output into the GDSII file. Other restrictions may also be applied, as defined in the GDSII Output setup.

Writing Parameterized Cells into GDSII File

Parameterized Cells (P-cells) are converted into ordinary cells during GDSII export. When an instance of a P-cell is written into a GDSII file, an ordinary cell is created based on actual P-cell parameters of this instance. The name of this cell is created by attaching actual values of parameters to the name of the P-cell.

As this may produce a long cell name, please take care of setting a sufficient limit for cell name length in order not to have truncated names for instances of P-cells in the output GDII file.

Notice also that P-cell instances with the same set of parameter values produce the same new cell name.

11.6: CIF I/O Setup

EXPERT may read/write CIF files for projects (**Project→Load...** and **Project→Save As...**) and cells (**Cell→Export..., Cell→Import...**).

You may set various options for CIF export/import from the **Setup→CIF Input/Output...** dialog, for input and output separately, on the corresponding pages of the dialog window.

CIF File Extensions

This field in the **Input** group inputs the list of possible extensions of CIF files in the following format:

```
*.cif;*.lay (a list of file specifications separated by semicolon) or *.* .
```

This list is used in the **Project→Load... and Cell→Import...** dialogs to filter the file list in the File Browser.

Only one extension may be specified for **output**.

Conversion Log

You may write all events during `.cif` file conversion into a log file. To do this, you must check the option **Write log file** and specify a file extension for the log file. For example, if you select the extension `.rep`, the messages during conversion of the file, `qs341.cif` will be written into the file, `qs341.cif.rep` (in the same directory).

11.7: Applicon I/O Setup

EXPERT may read/write Applicon files for projects (**Project→Load...** and **Project→Save As...**) and cells (**Cell→Export..., Cell→Import...**).

Various options may be set for the export/import of Applicon files from the **Setup→Applicon Input/ Output...** dialog, for input and output separately, on the corresponding pages of the dialog window.

Applicon File Extensions

This field in the **Input** group inputs the list of possible extensions of Applicon files as shown by the following example:

```
*.apl;*.apple;*.*
```

which is a list of file specifications separated by semicolon.

This list is used in the **Project→Load...** and **Cell→Import...** dialogs to filter the file list in the File Browser.

Only one extension may be specified for **output**.

Conversion Log

You may write all events during Applicon file conversion into a log file. To do this, you must check the option, **Write log file** and specify a file extension for the log file. For example, if you select the extension, `.rep`, the messages during conversion of the file, `qs341.apl` will be written into the file, `qs341.rep` (in the same directory).

If this option is checked off, such a file will be written anyway in the case of conversion errors or significant warnings, like a missing map for some layer or level, which might mean data loss.

Applicon Control File

This option allows you to select the name of a file with mapping of Applicon layer numbers to GDSII layer numbers/datatypes. Its syntax is very simple and clear from the example that comes with the distribution of EXPERT.

When a control file is absent or not specified, a warning is issued. In this case:

- import is performed anyway, with default layer names;
- export is not performed (i.e., a `CTL` file is obligatory for export into Applicon format).

Ignore Undefined Layers

If this option is on, the layers not defined in the specified map file will be skipped.

11.8: OASIS I/O Setup

EXPERT may read/write Oasis files for projects (**Project→Load...** and **Project→Save As...**) and cells (**Cell→Export...**, **Cell→Import...**).

Various options may be set for the export/import of Oasis files from the **OASIS Input/Output** panel, which can be displayed in the **Preferences** window (menu **Setup→Editor/Viewer**) then choose **OASIS Input/Output** in the **Import/Export Setup** group.

OASIS File Extensions

This field in the Input group inputs the list of possible extensions of OASIS files as shown by the following example:

```
*.oas;*.oasis
```

which is a list of file specifications separated by a semicolon.

This list is used in the **Project→Load...** and **Cell→Import...** dialogs to filter the file list in the File Browser.

The extension which is used for output is the first extension of the list of extensions.

Conversion Log

You may write all events during the OASIS file conversion into a log file. To do this, check the option **Write log file** and specify a file extension for the log file.

Output Options

Validate Scheme

From the drop-down list, you can choose to generate data of validation of the translation from EXPERT format to OASIS format (these data are stored in the last 4 bytes of the file).

- No scheme item means that no validation of the translation will be done.
- CRC32 validation: In this case a Cyclic Redundancy Check will be done using a 32 bits generating polynomial.
- Checksum32 validation: Using this scheme the validation data will be the sum of the ASCII values of all of the characters within the OASIS file truncated to its least significant 32 bits.

File Compression

Checking the **Compress** option will decrease the size occupied by cells and name tables in the OASIS file (using the “Deflate” compression method). Otherwise no compression will be done.

Relative Mode

This option concerns the coordinates of elements in the OASIS file. If checking the box, all (x,y) coordinates of objects will be stored relatively to the position of the last-written object of the file. Otherwise the absolute (x,y) position of elements will be written in the OASIS file.

Optimization Level

This feature can be used to choose the level of computing of the layout data in order to generate an OASIS file as small as possible. Three levels are available in the corresponding drop-down list:

- **None:** No optimization of size will be done. All points will be dumped using an arbitrary mode.
- **Medium:** In this mode of optimization, points are sorted and grid is computed (in arbitrary mode anywhere it is possible).
- **High:** The highest mode of optimization leads to building a sparse matrix then searching all patterns in this matrix (with gridded variants if possible).

11.9: Shortcut, Custom Menu, Custom Color Setups

11.9.1: Customizable Hotkeys

Setup→Shortcuts→Setup panel associates keyboard keys with EXPERT menu commands, so that the commands may be quickly executed by hitting the associated key instead of selecting them from menu. These associations are called **shortcuts, bindkeys,** or **hotkeys**.

You may assign EXPERT commands to almost all keys from the keyboard combined with **Shift**, **Ctrl** and **Alt** modifier keys. Notice however that combinations with the **Alt** key is the default way for Windows OS to navigate menus using underlined characters in menu command names. For example, **Alt E O <right arrow> W,** corresponds to the **Edit→Create Object→Wire** menu command.

Note: All functions of EXPERT menus can be assigned a keyboard shortcut, except in the following case. When several cells are opened in EXPERT, each of them can be accessed with a numbered item in the Windows menu, but no shortcut can be assigned directly to these items. In this case, you may switch between windows using standard Windows shortcuts: <ctrl-TAB> and <shift-ctrl-TAB>.

You may assign several shortcuts to the same command.

You may delete a shortcut.

You may restore EXPERT'S defaults for shortcuts.

To Assign a Shortcut

- select a command category: it corresponds to main menu items (including the commands for toolbars and dock-windows activation);
- select a command (it corresponds to an executable menu item);
- in the field **Press new shortcut key**, press the key combination you want to assign for this command (e.g., <Ctrl-Shift-F3>)
- click the **Assign** button. The name of this key will be added to the **Current keys** list.

Others Commands of the Shortcuts Panel

- You may remove a shortcut by select its command and its key in the **Current keys** list, and then click the **Remove** button.
- You may restore default shortcuts of EXPERT (see paragraph below)
- You may assign several shortcuts to one command by repeating the shortcut assignment many times.

Shortcut Setting File

Shortcut settings are saved in a file with the extension: `.ekb` using the **Setup→Shortcuts→Save Shortcuts** menu command. (The mnemonics for 'ekb' is '**E**xpert's **K**ey**B**oard'.)

Shortcut settings are loaded from a file with the extension `.ekb` using the **Setup→Shortcuts→Load Shortcuts** menu command.

Shortcuts Information

The **Setup→Shortcuts→Shortcuts Info** command shows a panel with the list of all shortcuts assigned to the keyboard. The shortcut panel has the **Save...** button that saves the hotkey list into a text file.

Reset Commands

The **Setup→Shortcuts→Default Shortcuts** command must be used when you want to restore the default shortcuts of the EXPERT installation directory for the current session.

The **Setup→Shortcuts→Clean Shortcuts** command must be used to delete all shortcuts in the current EXPERT session.

11.9.2: Toolbars Panel

The **Toolbars** panel allows to enable/disable existing toolbars, and it additionally allows you to define or customize the contents of these toolbars. This panel can be accessed with **Setup→Hotkeys/ Menus...→Toolbars** menu, or with **View→Toolbars→Customize** menu.

A toolbar can be enabled/disabled by checking/unchecking the corresponding box in the toolbar list.

Note: Toolbars (and dock windows) may also be enabled/disabled thanks to the contextual menu of the toolbars. This contextual menu may be displayed by a right-click on any displayed toolbar.

Toolbar Customization

- Select the requested toolbar in the list (it becomes highlighted in blue),
- Click the **Customize** button: the **Customize** toolbar panel is displayed (see Figure 11-18 below),
- All available buttons are listed in the **Available** toolbar buttons list with alphabetical order. Select any one and click the **Add** button in order to place it in the **Current** toolbar buttons list.
- Any of them can be removed from the **Current** toolbar buttons list by clicking the **Remove** button,
- Button order in the **Current** toolbar buttons list can be changed with the **Move up** and **Move down** buttons.

Note: The Design bar is the only toolbar which cannot be customized. Its shape and direction can be although changed in **Setup→Editor/Viewer→Editing** panel (see more details about it in Section 11.3.3: “Editing Page”).

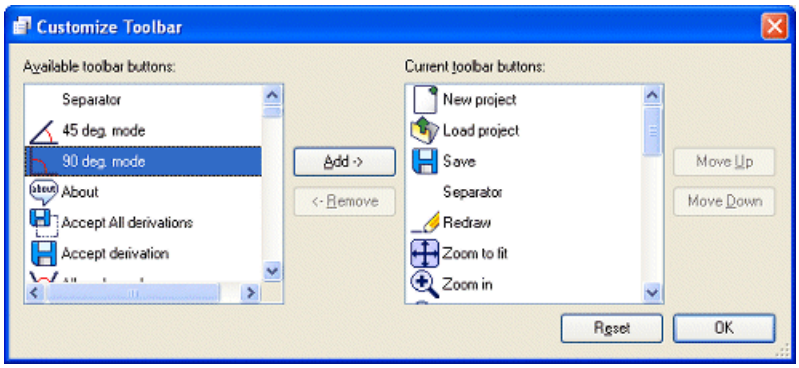


Figure 11-18: Customize Toolbar Dialog

Other Toolbar Characteristics

Each toolbar can be set to default state with Reset button of the Customize toolbar panel. The look of all icons of the toolbars can be modified in three ways:

- The icons size can be increased with the large icons option.

- The textual description of the function can be added in the icon thanks to display text labels option.
- Contextual description of the function can be displayed as tooltip (when holding the mouse on the icon) thanks to display hints option.

11.9.3: Application Colors Panel

In this setup panel it is possible to assign colors to various auxiliary drawings in EXPERT:

- cell map perimeter (active/inactive)
- cross line and cross hair cursor line
- layer and grid default color
- modified objects
- newly created object
- highlighted node
- selection and pre-selection markers
- cell origin and reference point
- ruler
- stretched shape elements
- router source and target (for selection, zoom, clip out, etc.)

Custom colors are set separately for black and white backgrounds.

It is possible to come back to colors defined in the current setup by clicking the **Set Current** button.

Colors can be chosen by selecting the desired item in the color category list, and then opening the select color panel (see Section "Layer Color Setup")

11.9.4: Custom Menu

If you click the right-mouse button inside the layout window, the custom menu appears. It contains a user-definable list of commands. You may put the most frequently used commands into the custom menu. Three different custom menus can be defined, depending of the tool which is active when you click the right-mouse button. Custom menus can be configured through the panel: **Setup→Hotkeys/ Menus...→Custom Menus** (see Figure 11-19).

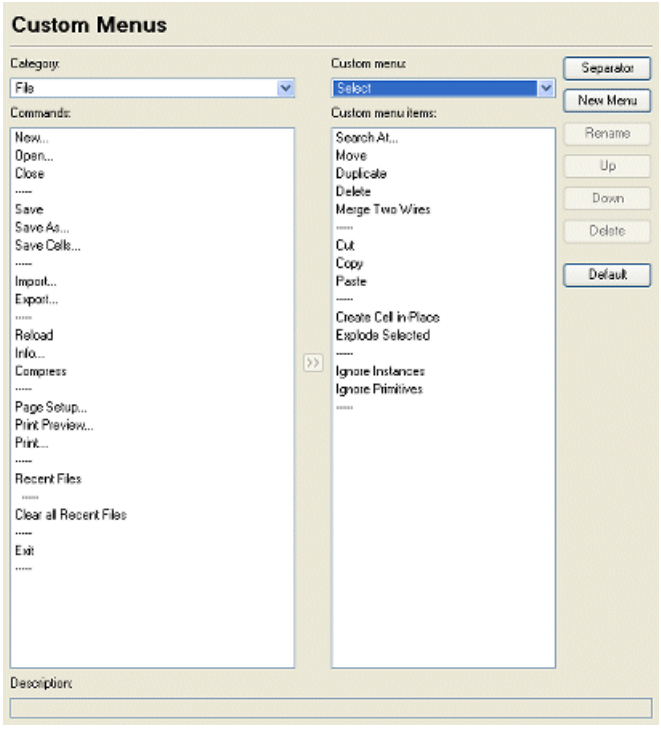


Figure 11-19: Custom Menus Panel

This dialog allows you to do the following operations.

Add an Item to Custom Menus

In the Category drop-down list:

- Select the EXPERT menu where you want to take the item to be added in a custom menu.

Note: Any function of any existing menu of EXPERT can be added to custom menus.

- Select an item from the **Commands** list. In fact, you may select a whole group of commands that correspond to a branch from the main menu. For example, if you select **Expert Clipboard** (from the Edit menu) in the Commands list, the following submenu will be added to the custom menu:
  - Cut
  - Copy
  - Paste
  - Paste into layer

At the same time, it is possible to add the commands 'Cut', 'Copy', etc., individually.

- In the Custom menu drop-down list, select one of the three available custom menus where you want to add an item:
  - Select
  - Stretch
  - System (i.e. all other cursor modes than select mode and stretch mode)

For example, if you add an item to the **Select** custom menu, the item will only appear in the custom menu when the cursor is in select mode.

- In the Custom menu items list, select the existing item below the place where you want to add the desired item. When you need to change the place of the new item in the list, select it and use the **Up** and **Down** buttons to move it.
- Click the '**>>**' button to move the selected item to the custom menu. You may add separators between groups of commands in the custom menu by clicking the **Separator** button.

Add a custom submenu to Custom Menus

It is possible to add an existing submenu to the **Custom Menu** (see the description above).

You can also create custom submenus in any of the three available custom menus. You can add any combination of items in these submenus, thanks to following sequence:

- Select the desired **Custom Menu** in the drop-down list.
- Click the **New** menu button and give the name of the submenu to be created. The new submenu is added in the **Custom Menu** items list, with an indented separator below indicating the end of the submenu.
- Add any item from the Commands list with the '**>>**' button, and place this item in the **Custom Menu** items list between the new submenu name and the separator, using the **Up** and **Down** buttons.

Other Functions of the Custom Menus Panel

- To remove an item from custom menu, select a line at the **Custom Menu Items** list and click the **Delete** button.
- To rename an item of custom menu, select a line at the **Custom Menu Items** list, click the **Rename** button and give the new item's name.
- To go back to default custom menu, click the **Default** button.

11.10: Built-in DRC Command Setup

The **Built-in DRC Command Setup** panel is used to set options controlling Built-in DRC (i.e. interactive DRC). These options are taken into account when a single DRC command is executed from the following commands of the **Verification→DRC** menu:

- Select
- Boolean operation
- Resize
- Layer
- Spacing check

The first four commands of this list can be used to generate layers thanks to boolean operations, or specific geometrical operations on drawing layers. The generated layers will be scratch layers, like those generated from a usual DRC script. The last **Spacing check** command will be used to perform a single geometrical check (spacing, distance, etc.) on any existing layer (drawing layer, derived layer or scratch layer generated by a previous interactive DRC command).

The available options of the **Built-in DRC Command Setup** panel allow following DRC operations setup:

- **Check tolerance** parameter allows you to modify error reporting (reduce the number of reported errors) for **DRC Spacing** checks of types LT, GT, RANGE and NE.
- The **Report at most <xxx> errors per check** option and the associated parameter allows you to restrict the number of errors reported by DRC checks. If this option is checked, the associated parameter defines the maximum number of errors to be reported by a DRC check command. This limit is introduced for DRC check commands because they can report a huge number of errors (the reasons may be various), and analyzing this number of errors becomes meaningless.
- The **Write log file** option is used for the generation of a log file containing information about the last DRC command which has been run (statistics about the amount of processed polygons, the amount of resulting errors and resulting scratch layer polygons). The format name of this log file is `<active_cell_name>.log`, and it is generated in the project directory.

If the **Append** option is checked, the DRC log file is appended to the existing file. Otherwise, the existing file is overwritten. The **Brief** option allows you to generate a short log file containing only the list of input polygons and the processing time.

- The **Scope** radio buttons control the amount of cell shapes which will be taken into account in the DRC processing: selected shapes or cell own shapes.

Note: Built-in DRC commands cannot be generated in hierarchical mode, in other words, the layers belonging to subcells of the active cell will not be taken into account. When you need to generate layers or perform geometrical checks as if the layout was flat, it is mandatory to write the corresponding commands in a DRC script and then to run this script with GUARDIAN DRC using the correct scope option. (see detailed description of the scope options in chapter 11.15 "DRC script run setup" and DRC syntax description in GUARDIAN USER'S MANUAL chapter 2.3)

- The **Saving layout** radio buttons allow you to save a project automatically, or be prompted to save the project before launching a DRC command. This can be useful in order to come back to the original project after having run the DRC command. Instead of deleting by hand the generated layers, use the **Cell→Discard changes** menu to restore the original layout.

11.11: DRC Script Run Setup

DRC Script Run Setup dialog is used to set options controlling execution of entire GUARDIAN DRC scripts, unlike the **Built-in DRC Command Setup** dialog (described above) which deals only with interactive DRC commands. It can be accessed by both menus **Setup→DRC Run** and **Verification→DRC→DRC run setup**.

The following options of this panel are found in the **Built-in DRC Command Setup** dialog panel (see Section 11.10: "Built-in DRC Command Setup"):

- **Check tolerance**,
- **Report at most <xxx> errors per check**,
- **Write log file**,
- **Saving layout and DRC script**,
- **Scope:** two radio-buttons are added regards to the **Scope** options of the **Built-in DRC Command Setup** dialog:
  - the **Cellwise** button will set the DRC processing to all shapes of all subcells of the current cell until the lower level of hierarchy. It means that the cross errors (that is to say errors caused by shapes of different cells) will not be detected by a DRC script if this button is checked.
  - The **Whole hierarchy** button will make the DRC script be processed on the whole contents of the current cell as if it was a flat layout. So the **Whole hierarchy** button must be set in the case when an exhaustive check of the geometry of the current cell is needed.

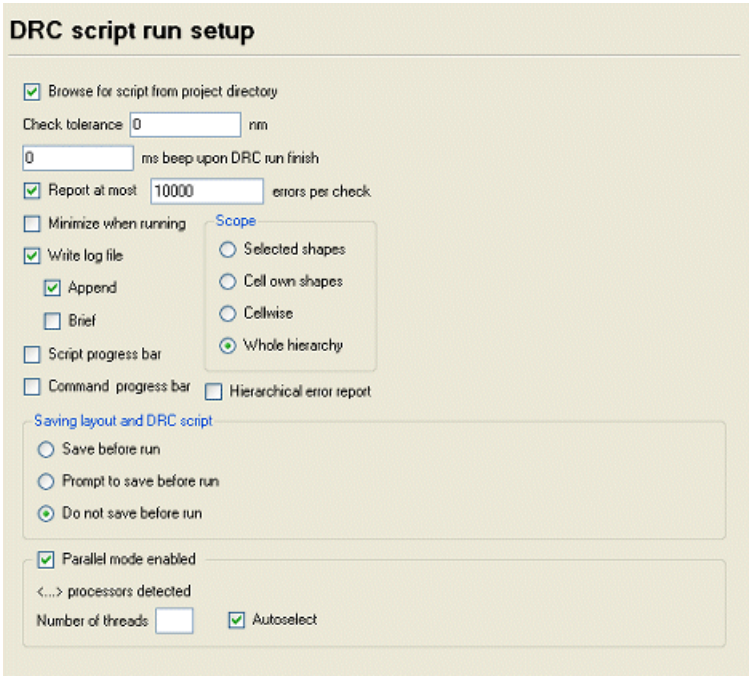


Figure 11-20: DRC Script Run Setup Dialog

Other options are specific to the **DRC Script Run Setup** panel:

- **Browse for script from project directory:** This box must be checked if the DRC script you are using is stored in the project directory. In this way, when loading a DRC script with the **File→Open** command of the DRC script panel (menu: **Verification→DRC→DRC script panel**), the browser window will be set to the current project directory. Otherwise, the browser window will point to last DRC script which has been loaded.

- **...ms beep upon DRC run finish:** If a nonzero value is entered into the edit field of the **... ms beep upon DRC run finish** setting, then DRC run started from EXPERT'S user interface will signal its finish by sound.
- **Minimize when running:** This option minimizes the EXPERT main window into the Windows taskbar, while the DRC script is executing. Once the execution of the DRC script is finished or terminated, the application's window is restored to its most recent state.
- **Script progress bar** and **Command progress bar:** These options are to enable the display the progress of script execution and the progress of the execution of every script command.
- **Hierarchical error report** allows defining output of error reports. This radio button can be activated in **Whole hierarchy** mode only. If this box is checked, `Guardian DRC` will recognize hierarchical structure of any error and produce a complete hierarchical error report (see detailed description in Section 2.4.2.2 "Buttons" of the GUARDIAN manual).
- Parallel DRC options:
  - **Parallel mode enabled** check box, to turn the mode on/off
  - **<..> processors detected:** Indicator of the number of parallel threads available
  - **Number of threads** input field
  - **Autoselect** checkbox

Their functionality correspond to Parallel_Threads DRC script command described in GUARDIAN manual Chapter 2 Section 2.3.5.12: PARALLEL_THREADS statement. GUI settings have precedence over the DRC script "Parallel_Threads" statement.

Note: The same panel as the DRC Script Run Setup can be accessed by menu **Setup→Current DRC script preferences** of the **DRC Script** panel (menu **Verification→DRC→DRC script panel**). In this panel, all options which are set are applied to current DRC script only, and have precedence on options activated through the **DRC Script Run Setup** dialog.

This page intentionally left blank.

Chapter 12: Tools

12.1: Layout Tools

This chapter describes tools to perform more advanced operations with layout (see Figure 12-1). The list below shows the essential tools used in a layout.

Measure submenu provides means for measurement on layout.

†**Xi script** submenu is for processing xi-commands.

†**Cross Sectional Viewer** controls the viewer of layout cross-sections.

†**Netlist Driven Layout**

†**Cut by Line** cuts objects into different parts. To choose how to cut the object, use **Cut by Line Modes**.

Cut by Vertices Count cuts wires and regions into smaller ones with a specified number of vertices.

†**Clip Out/ Clip In** is used to cut an area from the active cell layout and save it as a cell.

†**Merge Selected** joins the objects selected in a cell together.

†**Resize Selected** shrinks or expands the areas bounded by the selected objects in a cell.

†**Split** commands are used to split instance arrays, parametric wires, and multipaths into components: instances for arrays, boxes and wires for parametric wires, and wires and instances for multipaths.

†**Polygonize** submenu is for converting wires and circles to polygons.

†**Link Devices** commands allow you to automatically connect and align P-cell instances.

†**Scale Selected** is used to control the scaling factor in selected objects.

†**Scale Whole Layout** is used to control the scaling factor in the entire layout.

†**Snap Selected To Grid** is used to snap the selected objects to a grid.

†**Snap Whole Layout To Grid** is used to snap the entire layout to a grid.

Derived Layers is used to recalculate, generate, and clean derived layers.

Scratch layers menu is used to clean layers which are generated by a DRC script.

The **Tools** menu also contains other editing operations. See Section 12.16: “Various” for the functions of these operations.

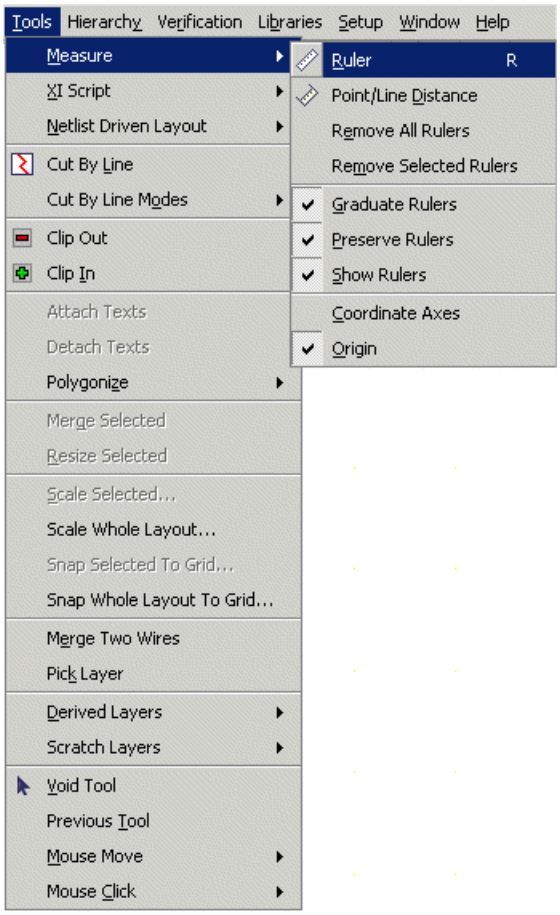


Figure 12-1: Tools Menu

12.2: Measurement

This section describes **Tools→Measure** submenu, also shown in Figure 12-1, and some other elements of EXPERT's interface related to measurements on the layout.

12.2.1: Metric Bar

The **Metric** bar (Figure 12-2) consists in two parts: Cursor Position (left part) and Measurement Units (right part).

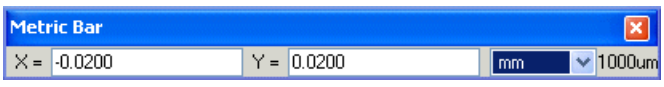


Figure 12-2: Metric Bar

Position Locator has two fields: **X-coordinate** and **Y-coordinate**. To define the coordinates of a point in the layout, place the cursor at the required point and read **Cursor Position** settings.

Cursor position may be **absolute** or **relative**. If no temporary reference point is activated, the positions are taken with respect to cell origin and called **absolute**. If a temporary reference point is activated, the positions are taken with respect to it and such positions are called **relative.**

The **Metric** bar indicates how to distinguish whether absolute or relative position is displayed by showing either "X=... Y= ..." or "RX=... RY= ...".

In addition, the **Edit→Position Cursor** command uses the position locator to pan the layout to the point specified by its coordinates, see Chapter 8: “Other Edit Submenu Operations”, Section 8.4: “Position Cursor”.

Information about distances and angles are displayed in the **Distance** field of the status bar (at the bottom of Expert window).

The **Distance** field is activated in the following cases:

- When the **Ruler** tool is used, the measured distance between two points, possibly along a polygonal line, is shown in the **Dist**. field. In the case of a polygonal line, the angle between the last two segments of the line is displayed (in degrees) in the **Ang.** field.
- When a temporary reference point is activated, the distance and the angle from it to cursor are shown.

The **Position Locator** and the **Distance** field show the values of the measurement unit set by you or assumed by default. The **Measurement Unit** field shows this unit and its value in microns. The **Measurement Unit** field is actually a drop-down list that allows you to select the measurement unit. You can set the measurement unit and the value (in microns) of **Lambda** (custom unit) by selecting **Setup→Editor/Viewer→General**.

Note: The number of decimals displayed by the **Position** and **Distance** fields can be set in the **Setup→Editor/Viewer→Viewing** panel in the **Decimals displayed** field.

12.2.2: Polyline Ruler

Purpose

Ruler determines the distance between two points, possibly along a polygonal path. In addition, it reports the angle, in degrees, between its two last segments.

The dialog tab **Setup→Editor/Viewer...→Ruler** allows you to set the ruler's font, color, and the style of ruler marking.

Execution

- To begin measuring the distance, select the **Tools→Measure→Ruler** menu command**.** The cursor assumes the **Ruler shape,** shown in Figure 12-3.

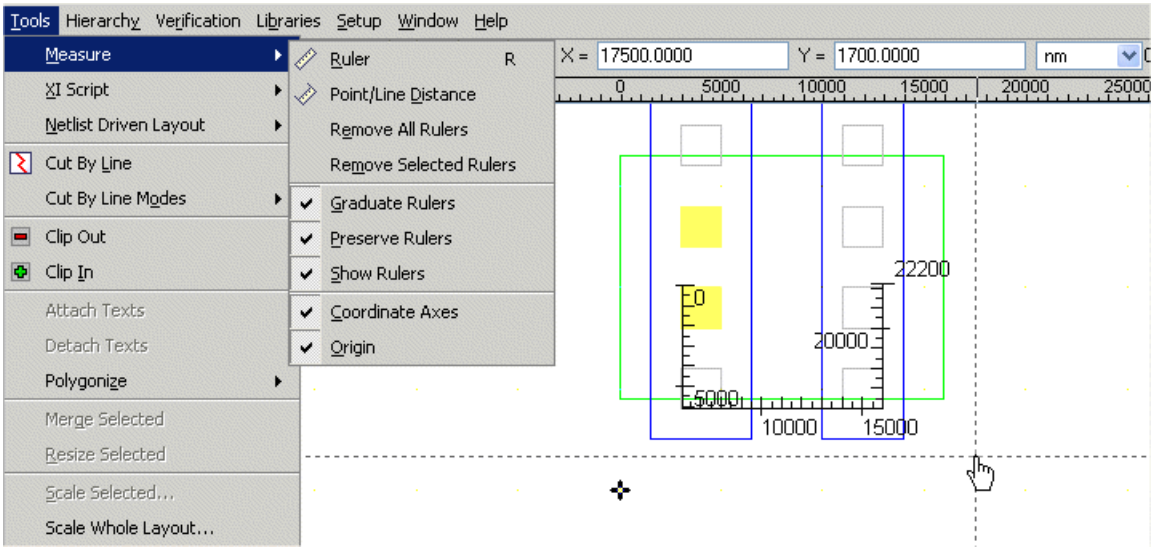


Figure 12-3: Rulers, coordinate axes, origin, cross-hairs

- To define the distance between several points, you must click on each of these points in order with the left-mouse button. This procedure is similar to drawing a polyline between these points. The length of this polyline defines the required distance.
- The precise distance is shown both in the **Status bar** and at the end of the last segment. At the same time the polyline can be graduated (see details of the ruler setup in the **Setup→Editor/Viewer→Ruler** panel described in Section 11.3.6: "Ruler Page").
- The **Status bar** shows also the **angle** between the two last segments, in **degrees**.
- To view the distance values directly on the ruler, check on the **Tools→Measure→Graduate Rulers** setting in the menu. The ruler is graduated with distance values in equal intervals; Otherwise, (the **Tools→Measure→Graduate Rulers** setting checked off) the value of the distance is shown only in the **Distance** field of the status bar.
- The last segment of a polyline ruler is done by clicking the right-mouse button. For a ruler to exist after finishing the distance measurements, the **Tools→Measure→Preserve Rulers** menu setting must be checked, otherwise the ruler disappears from the screen after finishing the distance measurements.
- The user can show/hide the rulers on/off the screen by checking on/off the **Tools→Measure→Show Rulers** menu item**.**
- The user can remove all existing rulers by clicking on the **Tools→Measure→Remove All Rulers** menu item**.** Otherwise, all created rulers will exist during the entire session of EXPERT.

- The user can delete some rulers by selecting them through the usual **Select→By Click/Drag** tool and by using the command **Tools→Measure→Remove Selected Rulers**.

12.2.3: Coordinate Axes

The user may switch on/off the display of coordinate axes by clicking at the **Tools→Measure→Coordinate Axes** menu item. If this menu item is checked on, then an X-coordinate axis is located on the top side and a Y-coordinate axis is located on the left side of EXPERT'S working area as shown in Figure 12-3.

The current cursor position is marked by thin lines on the coordinate axes.

12.2.4: Coordinate Origin

When the **Tools→Measure→Origin** menu item is checked, the origin is displayed (if it is contained in the display window). Otherwise (**Origin** menu item is unchecked), the origin is not shown.

12.2.5: Cursor Crosshairs

If the **View→Cursor Crosshairs** menu item is checked, all cursors in the layout editor are accompanied by two orthogonal dashed lines**,** shown in Figure 12-3. This helps to estimate the cursor position with respect to layout objects.

12.2.6: Distance from Point to Line

The command **Tools→Measure→Point/Line Distance** measures the distance between a point and a line. The line may be defined by a side of the boundary of a shape or by a segment of the centerline of a wire.

Execution

Click the point from which the distance is required, then pick a line segment to which the distance is required. When you try to pick a segment and move the cursor, you will see that segments close to the cursor are highlighted (like for the preselection). At the same time, the distance from the point to the highlighted segment is displayed at the end of the ruler. When you see the required segment highlighted, click the mouse second time to finish the measurement. After that an ordinary ruler will be drawn, starting from the point and orthogonal to the segment, see Figure 12-4.

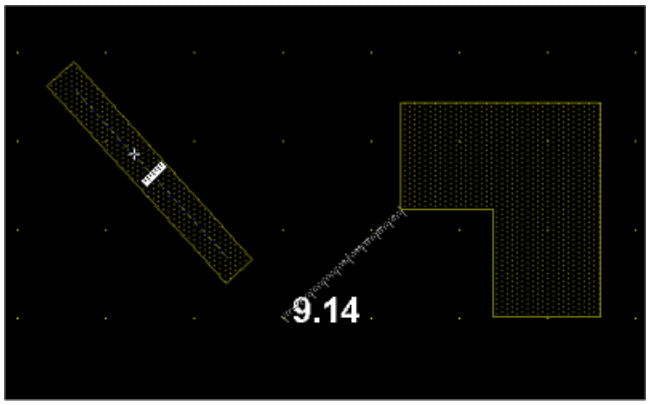


Figure 12-4: Examples of Point to line Distance

12.2.7: Distance from Reference Point

When a temporary reference point is activated (from the submenu **View** by **Activate Reference Point,** or **Toggle Reference Point** commands), the distance and the angle from the cursor position to the reference point are shown in the **Dist.** and **Ang.** fields of the **Status bar**.

12.3: †Xi Script Submenu

Only a brief description of the commands in this submenu are given here. The commands are described in detail in Chapter 17: "Expert Interface Scripts (XI-Shell)" Section 17.22: "Access to Xi-commands".

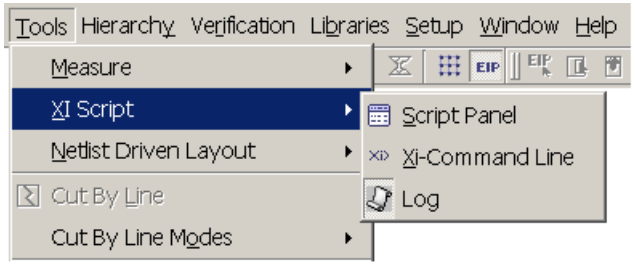


Figure 12-5: Xi-script Submenu

Script Panel command invokes the Xi Editor for editing and running xi-scripts.

Xi-Command Line enables the input of xi-commands from the xi-command line. This menu command places opens the Xi-Command Line dock window, if it was hidden, and places input focus of the keyboard in the edit field. The most convenient use of this command is via a bindkey. If the xi-command line dock window is visible, you may click the mouse inside its edit field to enable the input of xi-commands.

For details about the commands executable from the xi-command line, see Chapter 15, "**Expert Interface Scripts (XI-Shell)**".

Log command shows **Log** dock window which displays the xi-commands you use and the Expert responses during command execution depending on log filter specified on the **Setup→Editor/ Viewer→Data Safety** setup page. The following filters are available: prefix (output code), location (script line, only for xi-script execution), errors, warnings, commands, user responce, display (output of **LISA** command **DISPLAY**). The prefix identifies the type of output shown in the **Log** window. There are five types of output codes of log information.

- !E> Error message
- !W> Warning message
- !I> Information message
- !O> Output of **LISA** command **DISPLAY.**

12.4: †NDL/Netlist Driven Layout

Purpose

The command **Tools→Netlist Driven Layout** offers an additional tool to assist you in the creation of layout. EXPERT Netlist Driven Layout tool increases the productivity of layout design by automating cell generation and providing visual cues to assist the wiring process.

The Netlist Driven Layout is used to automatically create IC layouts based on information from a netlist. Cell instances are created from existing cells. P-cells are instanced with parameters from the netlist file. Netlist driven layout automatically transfers net names, instance names and connectivity information from netlist to the layout.

Figure 12-6 below illustrates the Netlist Driven Layout (NDL) submenu and its different functions.

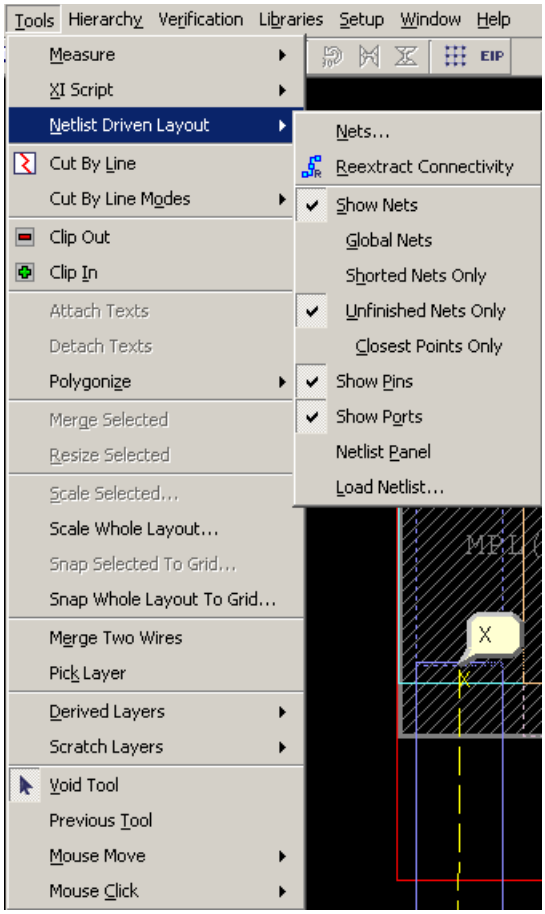


Figure 12-6: NDL Submenu

12.4.1: Loading the Netlist

In order to auto generate the layout based on the information from the schematic netlist, an existing ELD project file must be opened for the automatically generated cells to be placed.

Once the ELD project is open, you can choose **Tools→Netlist Driven Layout→Load Netlist...** submenu to load the desired netlist. A window titled **Choose netlist file** will open allowing you to make your selection of netlist.

After the selection is completed, if you press **Open** a netlist panel window titled **Netlist Editor - [path of the netlist file]** will open. This window can also be open by **Tools→Netlist Driven Layout→Netlist Panel**.

The Netlist panel (see Figure 12-7) can be divided in three windows:

- The **Netlist Rover** window, which help you browse through the netlist hierarchy.
- The **Windows** window, which list the netlist window currently opened.
- The **Netlist** window displaying the loaded netlist is illustrated in the center of Figure 12-7.

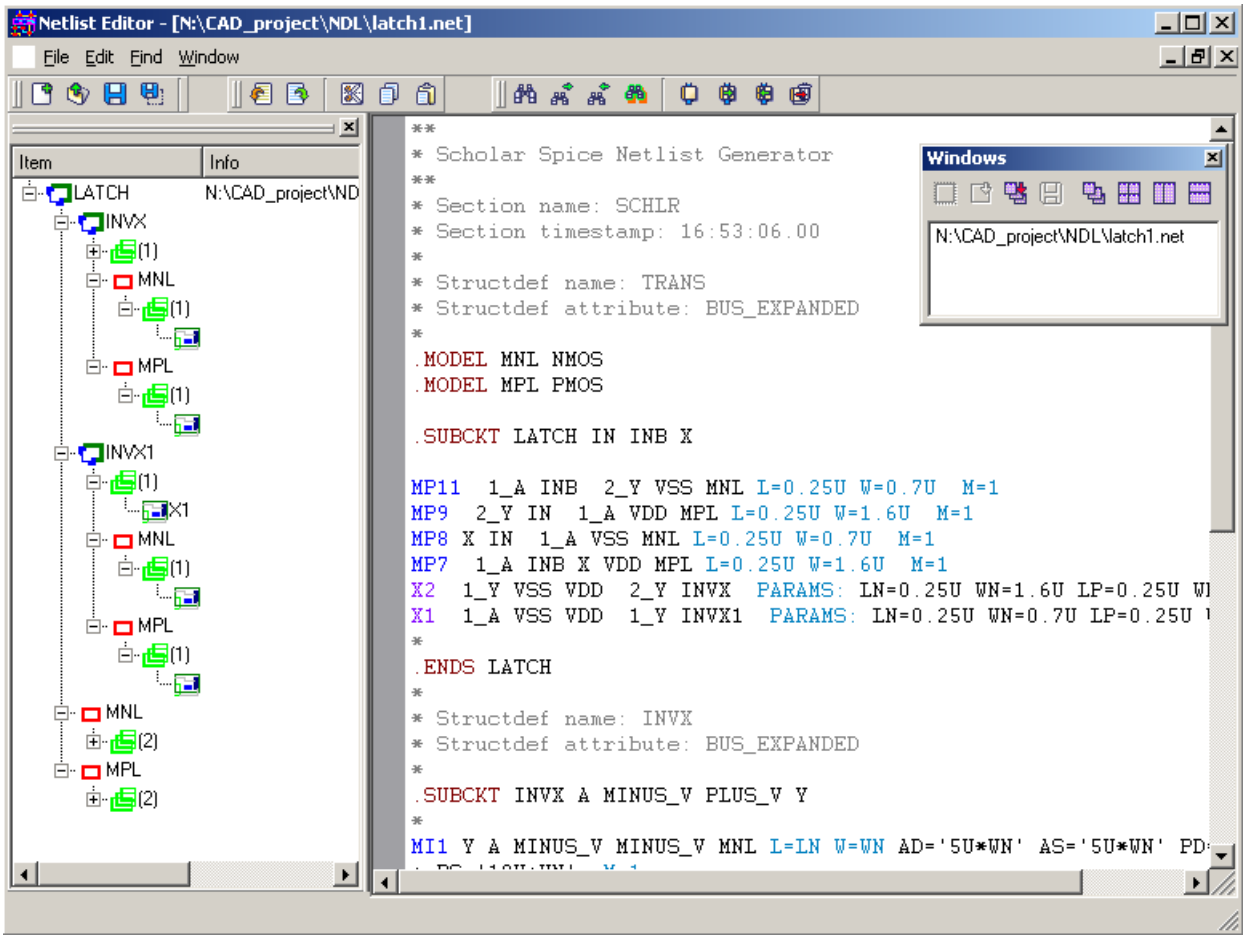


Figure 12-7: Netlist Panel

The **Netlist Rover**, showed to the right of Figure 12-7, illustrates the circuit hierarchical tree, allowing you to browse through and locate different instances/devices.

In order to run the Netlist Driven Layout tool, a library of predefined standard cells (resistors, capacitors, transistors, diodes, and other devices) must be present in your custom cell. These standard cells must be located in the current project.

For each cell or device referenced in the netlist there should be an equivalent cell or device in the ELD project.

The available cells can be P-cell. This will be described in more detail later in this section.

12.4.2: Cell Creation

Using the **Netlist Rover** window, you can generate the cells made of standard cells or P-cells available in your current project.

By placing the mouse cursor at the location of the subcircuit you want to lay out and clicking the right mouse button, a popup menu will appear (see Figure 12-8).

- Select **Create** in the popup menu and automatic cell generation will be performed.
- Select **Open** in the popup menu and the subcircuit selected will open in the layout window. If the subcircuit is non-existent in your standard cell or P-cell, you will be asked to create this layout cell. By pressing **OK** to this message, the automatic cell generation will take place.

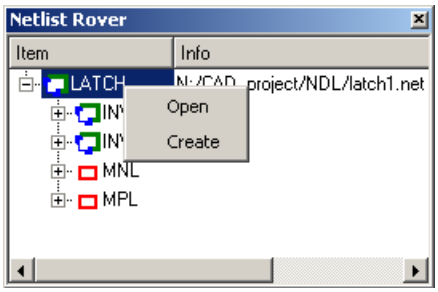


Figure 12-8: Create/Open Popup

Once the **Create** option is selected, the **Cell View** window displays the geometry of the created layout (Figure 12-9).

Note: If the selected subcircuit is hierarchical and the underlying instances are not calculated yet, they will be laid out as well.

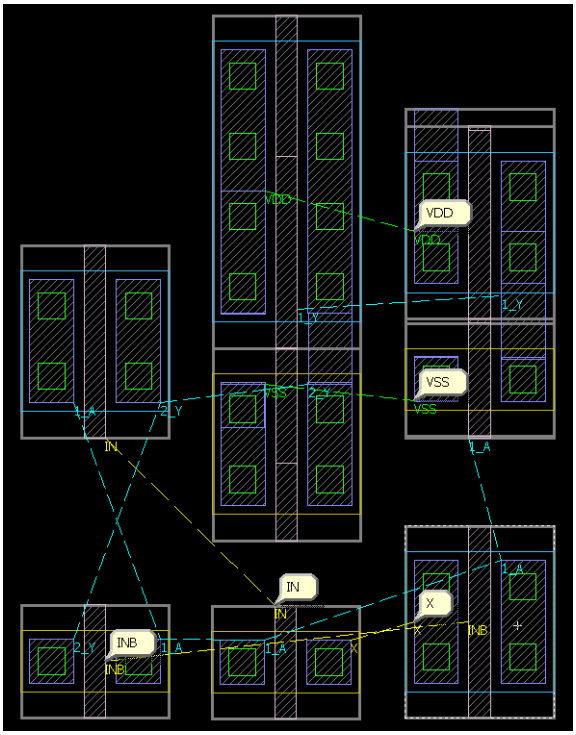


Figure 12-9: Initial Placement

12.4.3: Layout Editing

Once the automatic placement is completed, you can start editing the layout. To assist you in this step, EXPERT features a user-friendly interface for editing the generated design layout. To simplify the design wiring operation, electrical connectivity can be highlighted as displayed in Figure 12-9.

You can reorganize the initial placement of the instances and cells generated and the connectivity will update accordingly.

These highlighted lines are called flight lines. The purpose of these lines is to link the pins that belong to the same net and therefore should be electrically connected in the current cell.

There are three different types of flight lines: Global nets, Internal nets and External nets. The Global nets are displayed in green, the Internal nets are shown in blue and the External nets are displayed in yellow. The visibility of the flight lines can be switch on/off by **Tools→Netlist Driven Layout→Show Nets**.

12.4.4: NDL Submenu

The Netlist Driven Layout submenu contains many other useful features to assist you in the layout editing. This subsection describes them in more detail.

The **Tools→Netlist Driven Layout→Global Nets** option is used to hide or show Global nets. Global nets can have a lot of connections, therefore turning its visibility off decreases the congestion of flight lines and simplifies design editing.

The **Tools→Netlist Driven Layout→Shorted Nets Only** option indicates the nets that are mistakenly connected together (Figure 12-10). Shorted Nets are displayed in red.

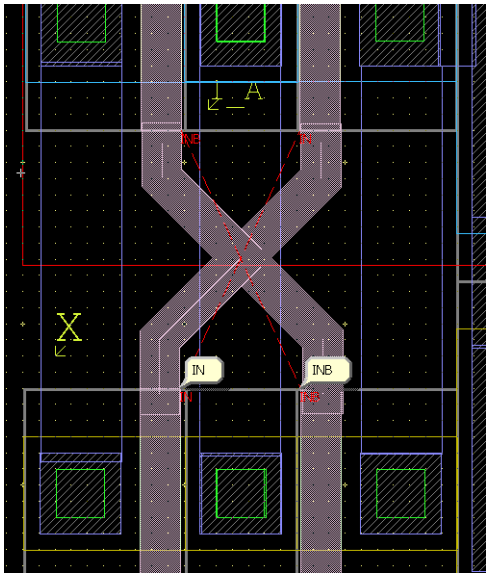


Figure 12-10: Shorted Nets Only

You can use **Tools→Netlist Driven Layout→Reextract Connectivity** or **Tools→Netlist Driven Layout→ Unfinished Nets Only** option to display nets that are not connected yet.

Tools→Netlist Driven Layout→Closest Points Only option display the flight line to indicate the closest point between two ports to be connected.

The **Tools→Netlist Driven Layout→Show Pins** option toggles the display of the pins.

The **Tools→Netlist Driven Layout→Show Ports** option toggles the display of the ports. The ports show pin geometry inside a cell instance. This is useful when the design is being edited in the “lazy” instance view mode. The ports are displayed with a dash blue line.

The command **Tools→Netlist Driven Layout→Edit Nets** invokes a window titled **Edit Nets** (See Figure 12-11), which allows you to add or delete nets, and change their type (Global, External or Internal). You can also modify the instance pin related to the selected net.

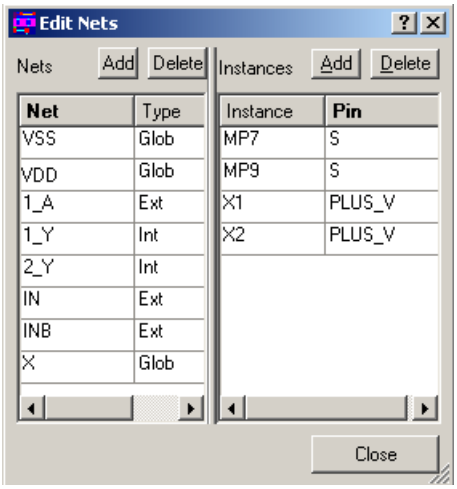


Figure 12-11: Edit Nets

The command **Tools→Netlist Driven Layout→Assign pins** assigns object pins from text labels, if connectivity between the text and objects layer exists, and the option **Generate ports** is set to on.

The command **Tools→Netlist Driven Layout→Update nets** turn on/off the automatic update nets flight-lines during layout modifications. For modes **Shorted Nets Only** and **Unfinished Nets Only**, this option automatically turns off (during modifications layout) because these options required connectivity information.

It this case, after turning this option on, you will receive a warning to update connectivity information.

12.4.5: Cells/P-cells Requirements

In order for the NDL feature to be able to establish the connectivity and display it with the use of flight line, nets and pins have to be specified in your standard cells or P-cells.

P-cells

In the project presented in this section, the cells MNL and MPL are actual P-cells. A portion of the script of the MNL P-cell is displayed in Figure 12-12.

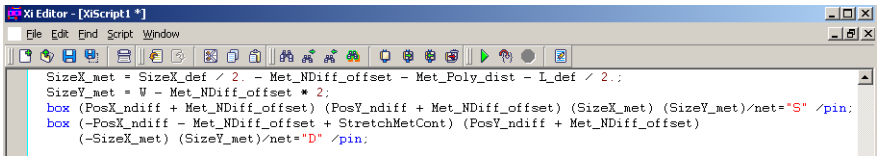

Xi Editor - [XiScript1 *]
File Edit Find Script Window
SizeX_met = SizeX_def / 2. - Met_NDiff_offset - Met_Poly_dist - L_def / 2.;
SizeY_met = W - Met_NDiff_offset * 2;
box (PosX_ndiff + Met_NDiff_offset) (PosY_ndiff + Met_NDiff_offset) (SizeX_met) (SizeY_met)/net="S" /pin;
box (-PosX_ndiff - Met_NDiff_offset + StretchMetCont) (PosY_ndiff + Met_NDiff_offset)
(-SizeX_met) (SizeY_met)/net="D" /pin;

Figure 12-12: MNL P-cell Script

In the script you can see that the text `/net= X` and `/pin` were added at the end of the box statement corresponding to the geometry representing the pin and net of the cell. The net `D` for drain and `S` for source are specified and will be use to establish connectivity.

Cells

For other cells, pins and nets can be specified through the property window.

Assuming you have chosen to create an inverter cell INVX from MNL and MPL P-cells, using the **Open/create Popup** window shown in Figure 12-8.

A Layout window will open to display the INVX inverter cell (see Figure 12-13).

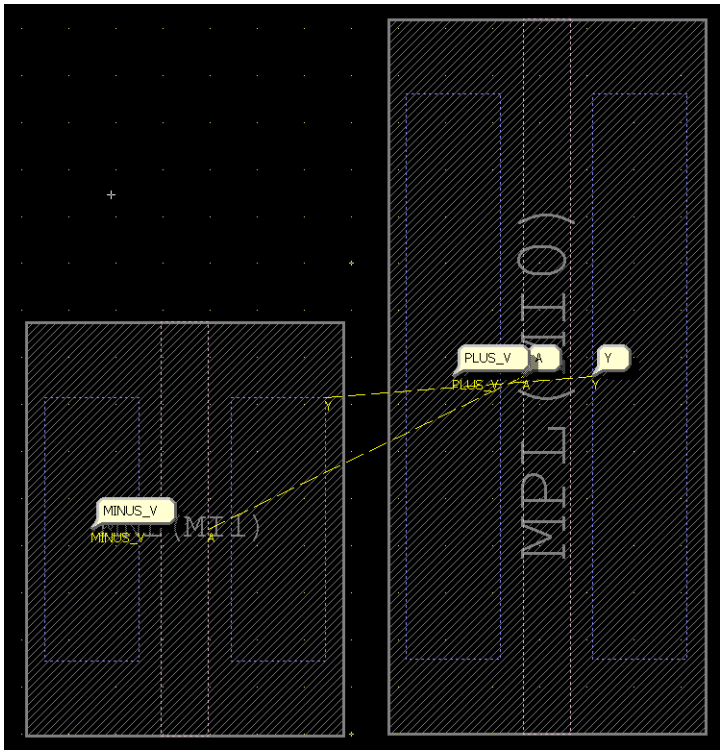


Figure 12-13: INVX Created

As a first step you can realign the inverter as shown in Figure 12-14.

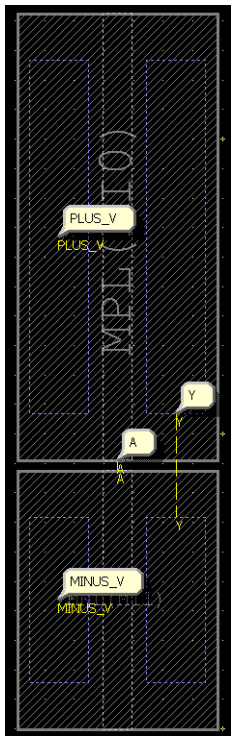


Figure 12-14: INVX Realigned

To create the pins for the net Y of the inverter illustrated in Figure 12-14, you can draw a metal rectangle to join both end of the net Y. Select this metal rectangle and modify its property as shown in Figure 12-15. By clicking in the **Net name** field, a drop down menu allows you to select the net name. In this case the user chose Y. If the **Is Pin** field is clicked, you need to change the state to "true".

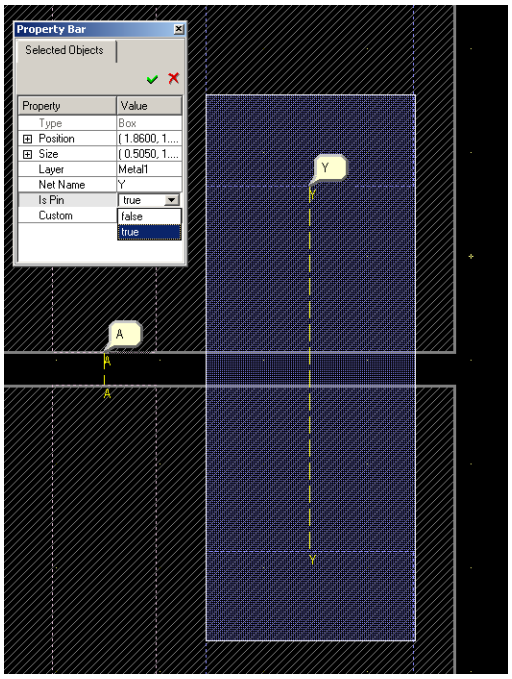


Figure 12-15: Pin and Net Name using Property Bar

Once these operations are completed, each instances of the INVX inverter cell will now display pins and nets as specified and the connectivity will be establish based on these.

12.5: Cross Sectional Viewer

12.5.1: Introduction

IC technologies continue to develop and become more and more complex. With increase of complexity in the contemporary design processes the designer must have knowledge not only of layout design, but in simulation and manufacturing as well. The more the design engineer knows about the process with respect to layout and technology, the higher the performance he can design.

Cross Sectional viewer is a tool within EXPERT to simulate the cross sectional views of ICs along an arbitrary drawn cut-line on the layout. Cross Sectional viewer is a link between the layout and the resulting device. It helps the designer to understand the spatial structure of the device being designed. Cross Sectional drawings are useful for understanding design rules, parasitic coupling and other design and fabrication problems.

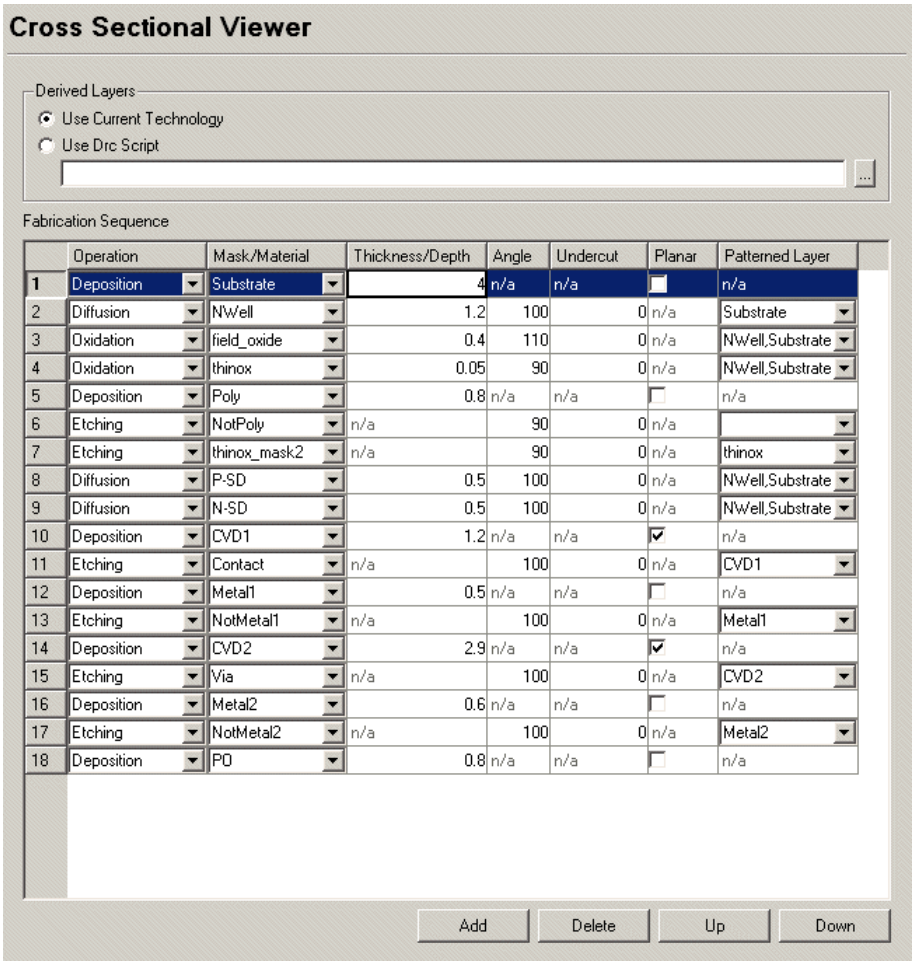
The current EXPERT implementation of the Cross Sectional viewer is based on the simplified simulation of the process steps using rough processing parameters from the EXPERT technology file.

12.5.2: Cross Sectional Viewer Setup

Approximate parameters for the simulation of the fabrication step must be specified in the technology file. For simplicity, assume that each fabrication step is associated with a separate layer in the layout. This layer may be either a drawn layer or a derived one; in other words, produced by boolean, resizing or any other DRC operations on the drawn layers.

There are two ways to create and modify process simulation parameters:

- Interactively (which is the recommended way), using the **Cross Sectional Viewer** setup dialog (see Figure 12-16 below) in the **Preferences** window. This panel can be accessed in two ways:
  - through the Preferences window (menu option **Setup→Editor/Viewer** then click on **Cross sectional Viewer** at the bottom of the panel list),
  - from the **Cross-Sections** window (menu option **View→Dock windows→Cross-sections**) then click the setup icon on the right side of the window.
- Manually, by editing the EXPERT technology file.



	Operation	Mask/Material	Thickness/Depth	Angle	Undercut	Planar	Patterned Layer
1	Deposition	Substrate	4	n/a	n/a	☐	n/a
2	Diffusion	NWell	1.2	100	0	n/a	Substrate
3	Oxidation	field_oxide	0.4	110	0	n/a	NWell,Substrate
4	Oxidation	thinox	0.05	90	0	n/a	NWell,Substrate
5	Deposition	Poly	0.8	n/a	n/a	☐	n/a
6	Etching	NotPoly	n/a	90	0	n/a	
7	Etching	thinox_mask2	n/a	90	0	n/a	thinox
8	Diffusion	P-SD	0.5	100	0	n/a	NWell,Substrate
9	Diffusion	N-SD	0.5	100	0	n/a	NWell,Substrate
10	Deposition	CVD1	1.2	n/a	n/a	☑	n/a
11	Etching	Contact	n/a	100	0	n/a	CVD1
12	Deposition	Metal1	0.5	n/a	n/a	☐	n/a
13	Etching	NotMetal1	n/a	100	0	n/a	Metal1
14	Deposition	CVD2	2.9	n/a	n/a	☑	n/a
15	Etching	Via	n/a	100	0	n/a	CVD2
16	Deposition	Metal2	0.6	n/a	n/a	☐	n/a
17	Etching	NotMetal2	n/a	100	0	n/a	Metal2
18	Deposition	PO	0.8	n/a	n/a	☐	n/a

Figure 12-16: Cross Sectional Viewer Setup

Derived Layers

The first field of the **Cross Sectional Viewer** setup is the **Derived Layers** radio button. Indeed, the derived layers can be useful in the process definition because several process steps are using masks, which are created thanks to boolean operations on other masks. This radio-button allows you to choose the derived layers which will be taken into account for the Cross-Section generation:

- If checking the **Use Current Technology** button, the derived layers which are integrated to the current project technology will be taken into account.
- If checking the **Use DRC Script** button, the derived layers will be generated from a DRC script. In this case, you are expected to type the path of this DRC script in the text field, or browse to it with

the browser button (the syntax of DRC scripts and the DRC script panel functioning are explained in GUARDIAN USER'S MANUAL).

Note: In any case, the derived layers must be in the technology of the current project. As a matter of fact, when using a layer of the technology for the process definition (either a physical layer or a derived one), it must be selected from a drop-down list. This list only displays the layers of the current technology.

Fabrication Sequence

The second field of the **Cross Sectional Viewer** setup is the **Fabrication Sequence** table. It allows you to define one-by-one all the steps of the process beginning with the bottom of the chip (substrate) to the top. Here are the available commands which can be performed on a selected process step (a process step can be selected by clicking on its number):

- **Add:** In order to add a process step, you must first select the step above the place where you want to add a process step. Then, click the **Add** button. A new step is added in the table; you can then set the parameters of this step. Each step is numbered automatically when adding it.
- **Delete:** In order to delete a step, first select it then click the **Delete** button.
- **Up and Down:** In order to change the place of a step in the fabrication sequence, **Up** and **Down** buttons should be used.

Process Operations

A process step is defined first by the operation it consists in. You can find below the detailed description of the process operations and the meaning of required parameters to define. There are four different kinds of process operations which are available:

- DEPOSITION
- ETCHING
- IMPLANTATION, DIFFUSION
- OXIDATION

For each operation, some parameters (or all of them) must be defined among the following list depending of the chosen operation:

- Mask/Material name
- Thickness/Depth value
- Angle value
- Undercut value
- Planar option
- Patterned Layers list

DEPOSITION

This operation is used to simulate metal deposition (or substrate creation) above the whole surface of the structure.

The available parameters and options to set for the deposition are:

- **Mask/Material** name: Defines the layer of the current technology which refers to this step of the process. The geometry of the layer along the cross-section does not matter for this step because a deposit is always done above the whole surface of the chip. The deposit will be done along the whole cross-section so that the layer can be empty or can contain some geometry.
- **Planar** option: This option was implemented to simulate "Chemical Mechanical Planarization", which has emerged as the best method of achieving planarity in semiconductor wafers following

such processes as deposition of dielectrics and metallization (for lithography to work properly, the wafer surface must be almost perfectly smooth). If this option is checked, the deposition will be planar. If not, the deposition will be conformal.

- **Thickness/Depth** parameter: the effect of this parameter will depend of the **Planar** option described above:
  - If the deposition is conformal (i.e., with a constant thickness on all the chip), this parameter will simply define this thickness of the material in current measurement unit (see Figure 12-17below).
  - If the deposition is set as planar (i.e., with a varying thickness along the cross-section depending of the relief of last deposit), the **Thickness/Depth** parameter will define the distance from the top last planar deposit to the top of the planar deposit which is being defined (the substrate is considered as the first planar deposit of the process description).

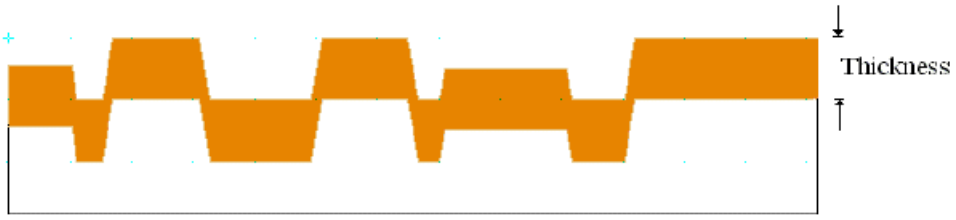


Figure 12-17: Thickness Parameter for Deposit Step

ETCHING

Etch operation simulates the etching or patterning of a material layer. The implementation of this process operation allows you to choose which layers will be potentially etched.

The available parameters and options for the etching step are the following:

- In the **Mask/Material** drop-down list, choose the layer whose geometry will define the mask of etching. It means that the etching will be potentially done under all geometries of the **Mask/ Material** layer. In short, this layer must represent the negative of the “real mask” used in chip fabrication (see Figure 12-18 below).
- In the **Patterned Layer** drop-down list, choose which materials will be etched (assuming that they are under **Mask/Material** layer shapes):
  - When selecting the empty field in this list, the etching will be done on any material at the surface of the device structure in the current state of the process.
  - When selecting the **Edit...** field of this list, the **Csv Patterned Layers** window will appear (see Figure 12-18). In this window you are prompted to select the list of materials which can be etched during this process step. (the **Shift** key can be used to select groups of layers and the **Ctrl** key can be used to deselect layers).

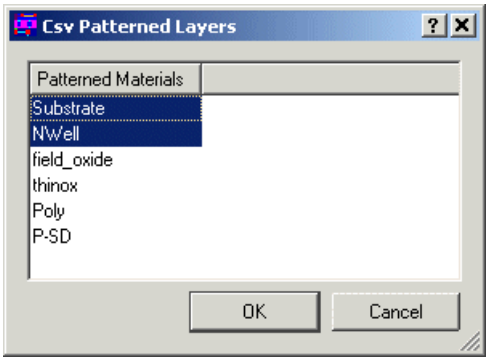


Figure 12-18: Csv Patterned Layers window

- The **Thickness/Depth** parameter defines the depth of the etching. It will limit the depth of etching to this value, even if there are several overlapping materials which can be potentially etched beyond this depth in regards to the **Patterned Layer** list.

  In other words, the selective etching allowed by the **Patterned Layers** can be limited in terms of depth by the **Thickness/Depth** parameter. If you want only a selective etching without influencing the depth parameter, set the **Thickness/Depth** parameter to a high value.

- The **Angle** value represents the slant effect of the etching in degrees. Set it to 90 degrees if you want to have vertical borders of etching for the current process step (see Figure 12-19).
- The **Undercut** value represents the shift between the border of the mask of etching and the real place where the etching begins (see Figure 12-19).

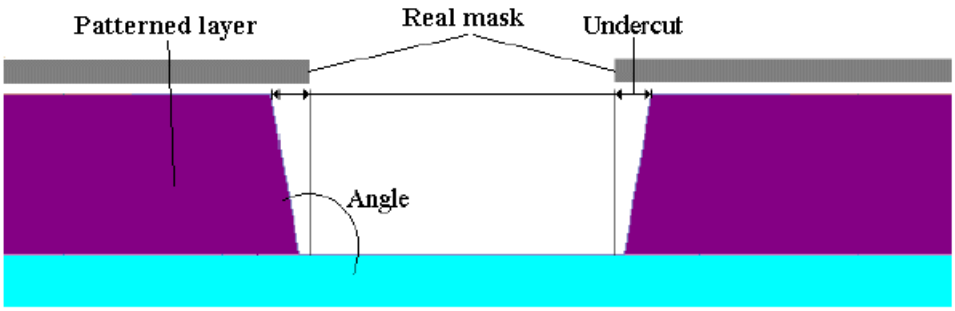


Figure 12-19: Etching Step Parameters

IMPLANTATION, DIFFUSION

Ion implantation and impurity diffusion processes are described using this step. These operations are used to create n+ and p+ doped regions, as well as n- and p-wells. For describing implantation and diffusion, the same model for etch is used, see Figure 12-19.

The available parameters and options for implantation and diffusion are:

- The **Mask/Material** drop-down list is used to define the mask layer of the implantation/diffusion. It means that the implantation/diffusion will potentially be done in the area corresponding to the geometry of the mask layer.
- The **Patterned Layer** field defines the list of materials where the implantation/diffusion will be done. It follows the same rules than for etching operation (see ETCHING above).
- The **Thickness/Depth** parameter defines the depth of the implantation/diffusion. It will limit the depth of implantation/diffusion to this value even if there are several overlapping materials which can be potentially implanted beyond this depth in regards to the **Patterned Layer** list.

In other words, the selective implantation allowed by the **Patterned Layers** can be limited in terms of depth by the **Thickness/Depth** parameter. If you want only a selective implant without influence of depth parameter, set the **Thickness/Depth** parameter to a high value.

- **Undercut** and **Angle** parameters can be used for simulating lateral effects that take place during high-temperature diffusion (see ETCHING above).

Note: In the resulting Cross-Sectional-View window the sections of materials representing an IMPLANTATION/DIFFUSION are displayed in “stipple” mode. (The stipple which is used is the same as the one for the display of the corresponding layer on the layout). This allows you to see clearly if several implantations are overlapping. All other materials (for deposit, oxydation..) are displayed in “solid” mode.

OXIDIZE

This step is to simulate local oxidation of silicon.

- The **Mask/Material** layer corresponds to the chosen material resulting from the oxidation.
- The parameter **Thickness** defines total thickness of the field oxide. It is assumed that field oxide both recesses into the surface (depth T1 = 0.45 * Thickness) and grows above the surface (T2 = Thickness - T1). See Figure 12-20.
- The **Patterned Layer** list allows selective oxidation in exactly the same way than the implantation/diffusion process step (see above). It means that the oxidation in patterned layers can be limited in terms of depth by the **Thickness/Depth** parameter.
- Using additional parameters, **Undercut** and **Angle**, you can simulate “bird's beak” effect (see Figure 12-20).

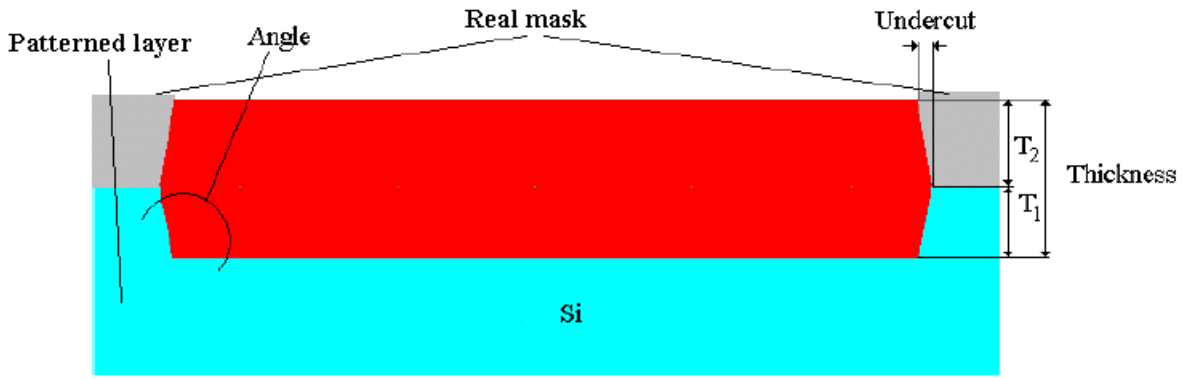


Figure 12-20: Local Oxidation Parameters

Note: Each process operation must be associated to a unique layer which will be the **Mask/Material** of the step. A layer of the technology cannot be used as **Mask/Material** for several process steps.

Technology File Syntax

As a result of the setup of the process description, all the defined data are stored in the technology of the current project. These technology data can be exported from the current project as an ASCII technology file (with menu **Setup→Technology→Save Technology file..**). This means that you can generate cross-sections on any layout if you loaded the layout with the technology file which is correctly set up for the **Cross Sectional Viewer**.

The other way of implementing the process description is to set the parameters by hand in the technology file for each layer corresponding to a process operation (see the description of the complete syntax of EXPERT technology file in Section 2.5.8: "Technology files (*.tcn file)"). This must be done by following the syntax of the example below:

```
Material
    {
    MaterialName = "VIA"
    …
    Thickness = 0
    }

Processing
    {
    DesignLayer = TRUE
    ProcessingStep = 15
    Operation = 2
    StepCoverage = 0
    Undercut = 0
    Angle = 100
    Planar = FALSE
    PatternedLayerName = "CVD2"
    }
```

In the technology file, the operation name is defined with a number thanks to the following code:

- 1: Deposit
- 2: Etching
- 3: Implantation
- 4: Diffusion (or "no process operation")
- 5: Oxidation

Note: In order to specify that a layer of the technology is not involved in any process step, set the `ProcessingStep` value to -1.

Note: The parameter `Stepcoverage` is unused in the current implementation of the Cross Sectional Viewer. The parameter `DesignLayer` can be left as it is by default.

12.5.3: Using the Cross Sectional Viewer

Cross Sectional Viewer requests specific information to be presented in the technology of the opened project. The **Setup→Editor/Viewer** then **Cross Sectional Viewer** item pops up the **Cross Sectional Viewer** setup panel for adding and editing the parameters of the simulation of fabrication steps. See Section 12.5.2: "Cross Sectional Viewer Setup" for detailed information about the process setup.

Besides the setup panel, the Cross Sectional Viewer consists in two separate dock-windows for its current use:

- The **Cross Sections** dock-window is the command window (see Figure 12-21). It can be used first to generate new cross-sections on the active cell of the layout and secondly to activate existing cross-sections of the layout. This window can be displayed thanks to the menu **View→Dock Windows→Cross Sections**.

- The **Cross Sectional View** window is the output window (see Figure 12-22). It is the window where cross-sections are displayed and where all the viewing commands of the Cross Sectional Viewer can be activated. This window can be displayed thanks to the menu **View→Dock Windows→Cross Sectional View**.

These two separate windows are closely linked to each other in the current use of the Cross Sectional Viewer: they can be displayed in one shot thanks to the menu **Tools→Cross Sectional Viewer**. Moreover, they can be docked to the GUI for an easier use (see Figure 12-23).

Cross Sections Generation

To generate a Cross Sectional view for the opened cell:

- Click the **+** green button of the **Cross Sections** window.
- Specify a cut-line in the active cell of the layout. You must move the cursor to a point on the layout view, press and release the left button, drag a line across layout in an arbitrary direction, and finally press the left button to finish the cut-line.
- After drawing the cut-line, the processing is automatically launched. The **Cross Sectional View** window displays the calculated cross-section view, and the coordinates of starting point and ending point of the cross-section are displayed in the Cross Segments list of the **Cross Section** window.

Repeat this step as many times as is needed in order to create several cross-sections. In such a case, you can display any of these cross-sections by clicking on its starting point coordinates in the **Cross Segments** list. The cut-line is then highlighted in white on the layout. Other cut-lines are highlighted in red (see Figure 12-23).

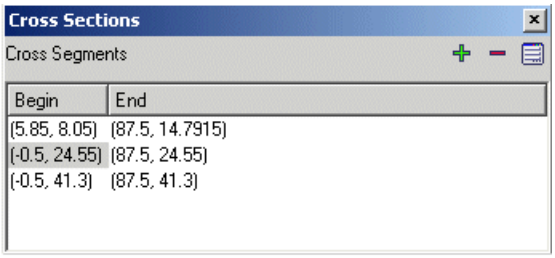


Figure 12-21: Cross Sections window

An alternative way to generate a Cross Sectional view for the current cell is to use the following XI script command through the command line of EXPERT (see Figure 12-23):

```
'cross view (start_x) (start_y) (end_x) (end_y)'
```

This kind of command can be included inside a script and executed in EXPERT. The detailed description of the EXPERT XI script commands (LISA language) can be found in Chapter 17: “Expert Interface Scripts (XI-Shell)”.

Note: When defining the cut-line on the layout, the cursor is snap mode sensitive. It means that you can activate any mode in the menu **Edit→Direction Snap Mode** in order to choose the direction of the cut-line (Horizontal, Vertical, No Direction...)

To remove a Cross Sectional view:

- Select it in the Cross Segments list of the **Cross Sections** window.
- Click on the **-** red button of the **Cross Sections** window.

Cross Sections Visualization

When generating a cross-section, it appears in the **Cross Sectional View** window (see Figure 12-22).

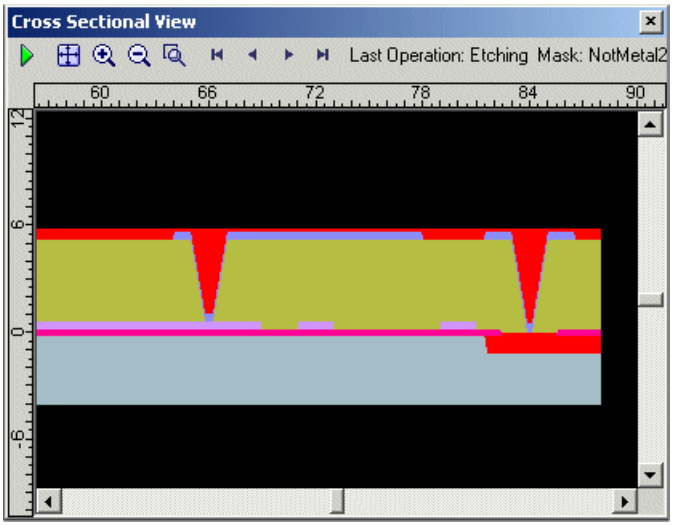


Figure 12-22: Cross Sectional View Window

Several features are available in this window in order to easier the visualization:

- **Coordinate axes** are implemented in order to check the distances. The graduations unit follows the setup of the metric bar of EXPERT.
  - The vertical axis corresponds to the depth in the chip. Its origin corresponds to the top of the first deposit (substrate).
  - The horizontal axis corresponds to the distances along the cut-line. Its origin is set at the start point of the cut-line.
- **Zoom:** Four buttons are available for the zoom functions - **fit to Cross Sectional view**, **zoom in**, **zoom out** and **zoom into window**. These functions can be used with the hotkeys which are active in EXPERT for the corresponding functions of zoom on the layout.
- **Step-by-step view:** The **Cross Sectional Viewer** has a feature that allows you to view the actual individual fabrication processes one step at a time and in the exact sequence that they occur. There are four "blue arrows" buttons managing this step-by-step view in the **Cross Sectional View** window:
  - **First Operation** and **Complete Sequence** buttons will respectively display the cross-section state after the first operation and after the last operation of the fabrication sequence,
  - **Previous Operation** and **Next Operation** buttons allow seeing respectively the cross-section before the current step view, and after the next process step of the fabrication sequence. The current process step is described on the right side of the step-by-step buttons thanks to the **Material** name and the **Operation** name of the last processed operation.
- **Update Cross Sectional view:** It can be useful to update a cross-section regards to some changes in the layout or in the process parameters. By clicking the **Update** button (green arrow) in the **Cross Sectional View** window, you will regenerate the selected cross-section without having to redraw the cut-line.
- **Show/Hide material labels:** This button is just at the right of the **Run** button. It can be used to display or hide pop-up labels containing the name of each material of the cross-section. These labels are automatically centered on the material they are pointing to (whatever the zoom is).
- **Ruler:** The ruler tool is available in the **Cross Sectional View** window, and can be activated exactly in the same way for the layout. (The corresponding hotkey can also be used to activate this

ruler). In order to delete rulers after having done the measurements, only the command menu **Tools→Measure→Remove all rulers** can be used.

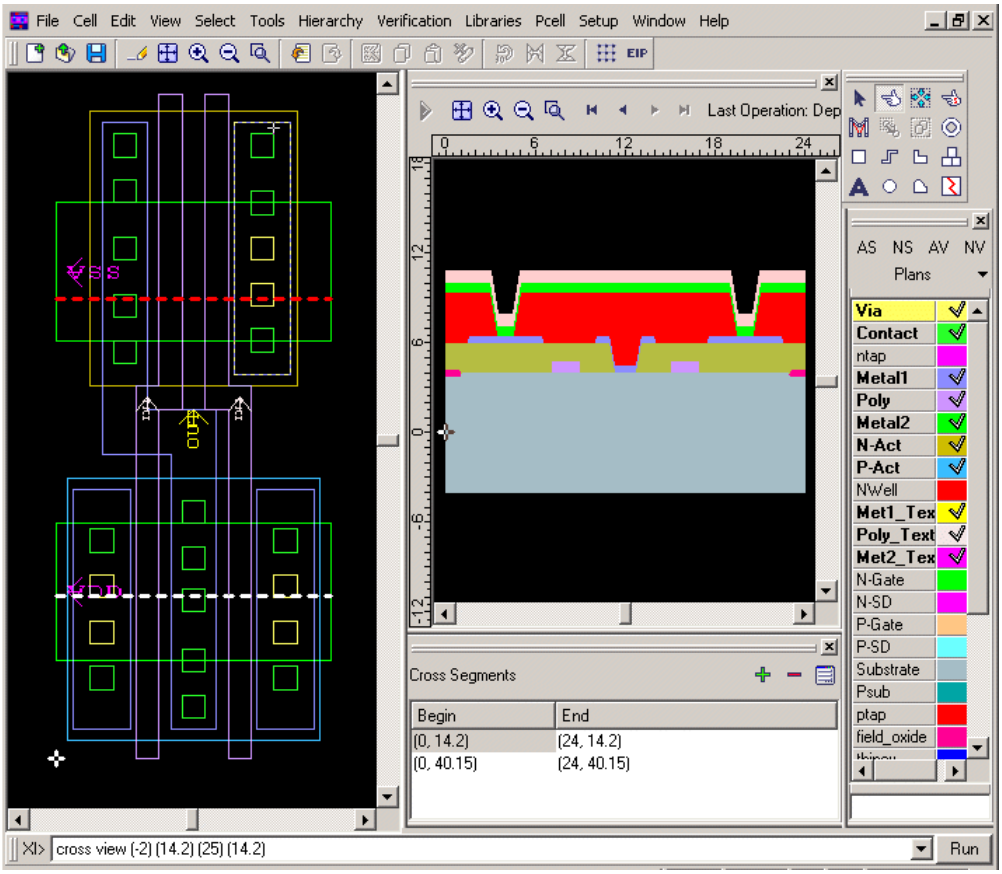


Figure 12-23: Cross Sectional Viewer Example

12.6: Device Linking

The EXPERT Device Linker is a tool that allows you to automatically connect and align two instances (devices) with pins connected to the same net. In the current implementation, only the instances of the same parameterized cell masters can be linked.

12.6.1: Linking Setup

To set up P-cells for linking, add the following components (see Section 17.20.3: “LAYOUTOBJECT Type”) for the pin objects: linking access direction (component `linkdir`) and reference to the user defined linking procedure (component `linkcallback`). For example:

```
_pin = (box 0 0 5 5 /Layer = "Met1" /pin /net="S" /create /get);
_pin.linkdir = DLD_LEFT;
_pin.linkcallback = "mos_linker";
```

The following constants `DLD_LEFT`, `DLD_RIGHT`, `DLD_TOP`, `DLD_BOTTOM` are predefined to set up linking access direction. Only pins with opposite directions can be linked. The linking directions are properly updated if instance rotated and/or mirrored. User defined linking procedure is to adjust the parameters of linking P-cell instances. The EXPERT Device Linking tool provides the following arguments for linking procedure:

- The P-cell parameters of the first instance being linked in the form of case sensitive LISA ASET object, where the name of each attribute corresponds to the name of the P-cell parameter, and the value of each attribute corresponds to a P-cell parameter value.
- The P-cell parameters of the second instance in the form described above.
- The pin name of the first instance. The pin name coincides with the name of net to which it belongs.
- The pin name of the second instance.
- The boolean value FALSE, if pins are only pins in the external net; TRUE if there are other pins connected to the same net.

The linked procedure is invoked automatically when all linking conditions are satisfied. When the linked procedure is finished, the new P-cell parameters are applied to the linked instances, and the last ones are aligned along the edges where linking had taken place.

12.6.2: Automatic Linking

Automatic linking allows newly created objects to be automatically linked with existing objects, if their pins overlapped and other linking conditions are satisfied. An object for linking can be the result of any of the following editing operations: create, modify, move, duplicate, rotate, mirror or paste.

The following conditions must be satisfied for automatic linking:

- Turn On the Automatic Device Linking option (**Setup→Editor/Viewer→Editing**).
- Define the linking procedures by running corresponding xi scripts, or include their definitions into start up xi script.
- Both instances must have same P-cell master and set up for linking (see set up procedure above).
- Overlap instance pins with the last ones connected to the same net.
- Make sure that pins have proper access direction and the same linking procedure.

Example

```
! p-cell definition with added linking parameters
DEFINE PCELL "nmos" /REPLACE
  PARAMETER Width /TYPE = (Integer) /DEFAULT = (5)
  PARAMETER Length /TYPE = (Double) /DEFAULT = (1.0)
  PARAMETER LeftCont /TYPE = (Boolean) /DEFAULT = (true)
  PARAMETER RightCont /TYPE = (Boolean) /DEFAULT = (true)
  BODY BEGIN
  …
  ! add linking parameters
  pwire {(_x), (_y1), (_x), (_y2)} /width=(_wwidth)
            /layer=(_metalLayer) /flushend /net=(_s_net) /pin
            /boxes = ({{attr_create(PWA_LAYER, _contLayer),
                       attr_create(PWA_WIDTH, _contWidth),
                       attr_create(PWA_LENGTH, _contWidth),
                       attr_create(PWA_SPACE, _cont2contSpace),
                       attr_create(PWA_DISTRIBUTION, BDT_MIN),
                       attr_create(PWA_BEGIN_OFFSET, -_metalOverCont),
                       attr_create(PWA_END_OFFSET, -_metalOverCont)}});
  _leftPin= (wire {(_x), (_y1), (_x), (_y2)} /width=(_wwidth)
             /layer=(_metalLayer) /flushend /find /get);
  if (_leftPin NEQ NIL) then begin
    _leftPin.linkdir = DLD_LEFT;
    _leftPin.linkcallback = "nmos_linker";
  end;
  …
```

```
END;

!linking procedure
define procedure "nmos_linker" /replace
  parameter device1Params
  parameter device2Params
  parameter device1Pin
  parameter defice2Pin
  parameter externalConn
  do begin
    if (device1Pin EQL "D") then begin
       device2Params["RightCont"] = false;
       if (not externalConn) then begin
         device1Params["RightCont"] = false;
       end;
    end;
    if (device1Pin EQL "S") then begin
       device2Params["LeftCont"] = false;
       if (not externalConn) then begin
         device1Params["LeftCont"] = false;
       end;
    end
  end;
```

The results of linking two instances of the P-cell `nmos` depicted in Figure 12-24. One of the instances (the lower one) was rotated 180 degree to make proper linking directions. The rotated instance was moved into direction to first instance while pins overlapped. Once pins are overlapped, the EXPERT Automatic Linker is triggered. Because there are no more pins connected to the external net, the linking procedure `nmos_linker` is invoked with parameter `externalConn=false`, and no contacts between the two linked instances are presented.

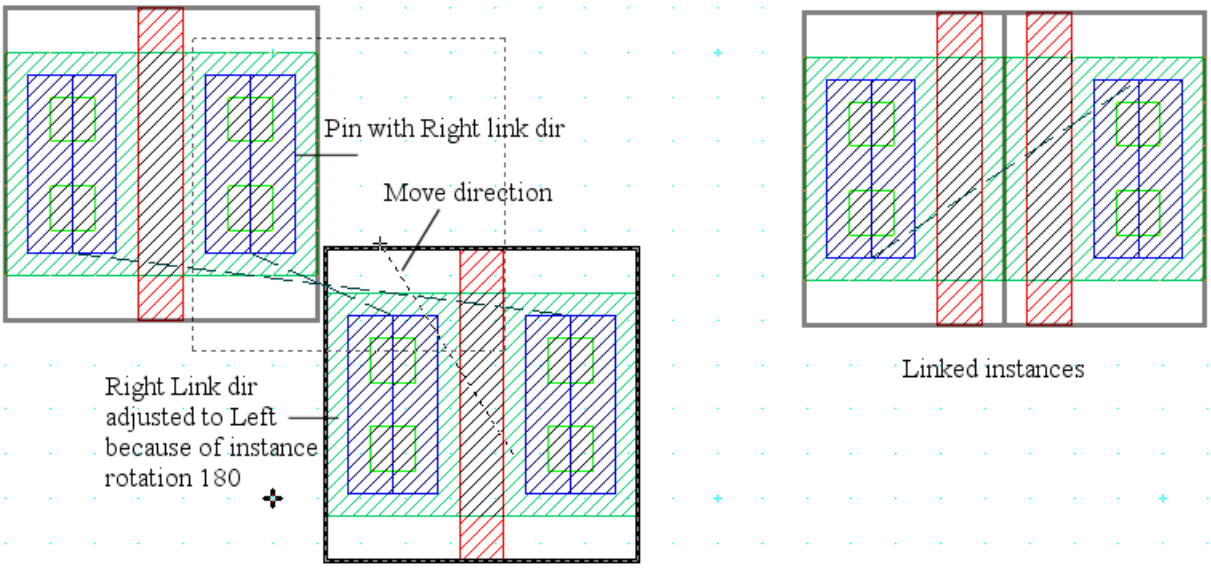


Figure 12-24: Automatic Instance Linking

When the pins of two linked instances are connected to other pins (see Figure 12-25), the contact between them is presented.

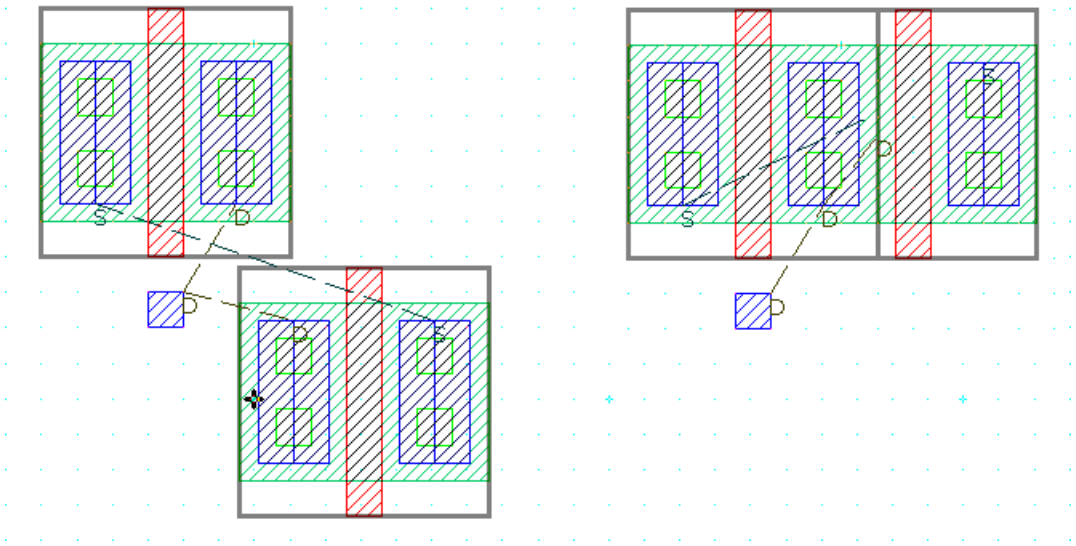


Figure 12-25: Automatic Instance Linking with External Connection

12.6.3: Manual Linking

The command **Tools→Link Devices→Link** allows preliminary selected instances (at least two instances should be selected) of the same P-cell to be linked into a chain. Their P-cells should have all necessary attributes for linking (pins with specified linking directions and references to linking procedures). The linking procedures must be defined before executing the **Link** command. The linking directions for the selected instances are adjusted automatically by appropriate rotation. For example, if EXPERT Device Linker accepts two instances with pins connected to the same net, but with invalid directions for linking, then one of the instances (not linked one) is rotated to match the linking access direction. Figure 12-26 shows transistors before (first row) and after linking (second row).

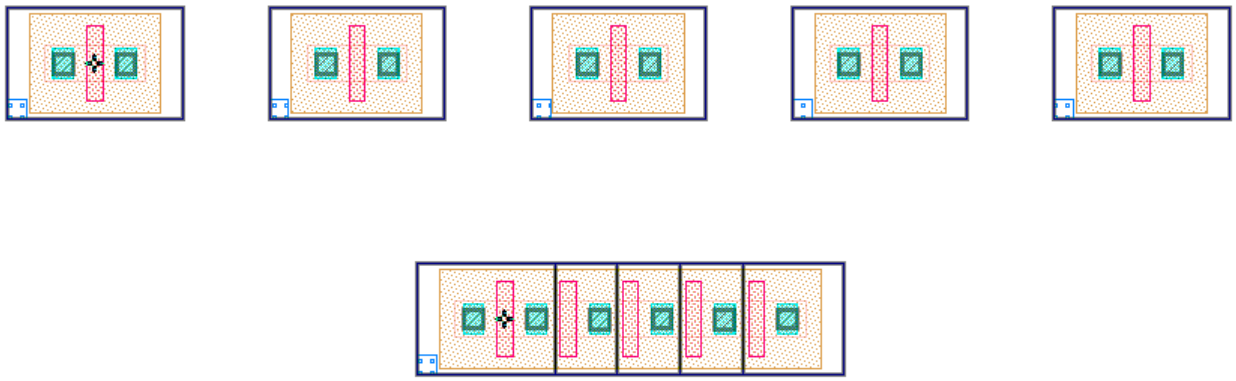


Figure 12-26: Chain Instance Linking

To unlink instances, select a linked group by selecting any instance within a group and execute the **Tools→Link Devices→Unlink** command. The EXPERT Device Linker restores the parameters of the instances before linking, but positions before linking are not preserved. The unlinked instances are left in the same positions (linked state), and might be overlapped (see Figure 12-27).

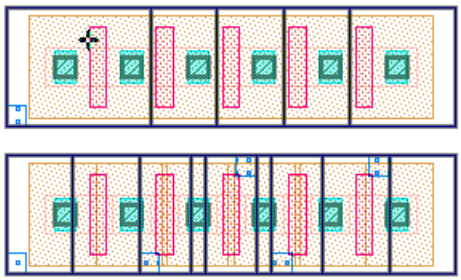


Figure 12-27: Unlinked Instances

The command **Tools→Link Devices→Unlink All** will unlink all instances in the current cell.

12.7: †Cut by (Poly)Line

Purpose

To cut primitive objects into one or more parts.

The goal of cutting primitive objects into one or more parts is to remove a piece of a geometrical object or to simplify its shape, etc. Several objects may be cut simultaneously.

There are three modes for cut by line, selectable from the **Tools→Cut by Line Mode** submenu:

- cut in the active layer;
- cut selected objects;
- cut all;
- save wires.

The first three self-explanatory modes define which objects will be cut.

If the option **Tools→Cut by Line Modes→Save Wires** is on, then the cut-by-line operation does not convert wires into polygons. Wires are split into several wires by points of intersection of its centerlines with the cutting line. Wire styles are preserved. In particular, if the original wire has the "Extended" end style, then the resulting wires will overlap near the splitting point.

Execution

- To start cutting objects, click the **Cut By Line** button at the Design Bar or select the **Tools →Cut By Line** menu item.
- To cut object(s), draw a polyline. This polyline will be a line along which the object(s) is(are) cut.
- In the message line, the angle between two last segments is shown in percents and degrees. This is useful, for example, when cutting a donut into segments.

Results

All primitive layout objects that are subject to cutting according to the current cut mode, and intersected by the polyline "from boundary to boundary", are cut into two or more parts.

- Pieces of the polyline being drawn outside the object(s) does not affect any of the objects.
- A "hanging" piece of polyline (i.e., if an end of the polyline is inside an object) produces no cut (see Figure 12-28).
- Cut objects are converted into regions.

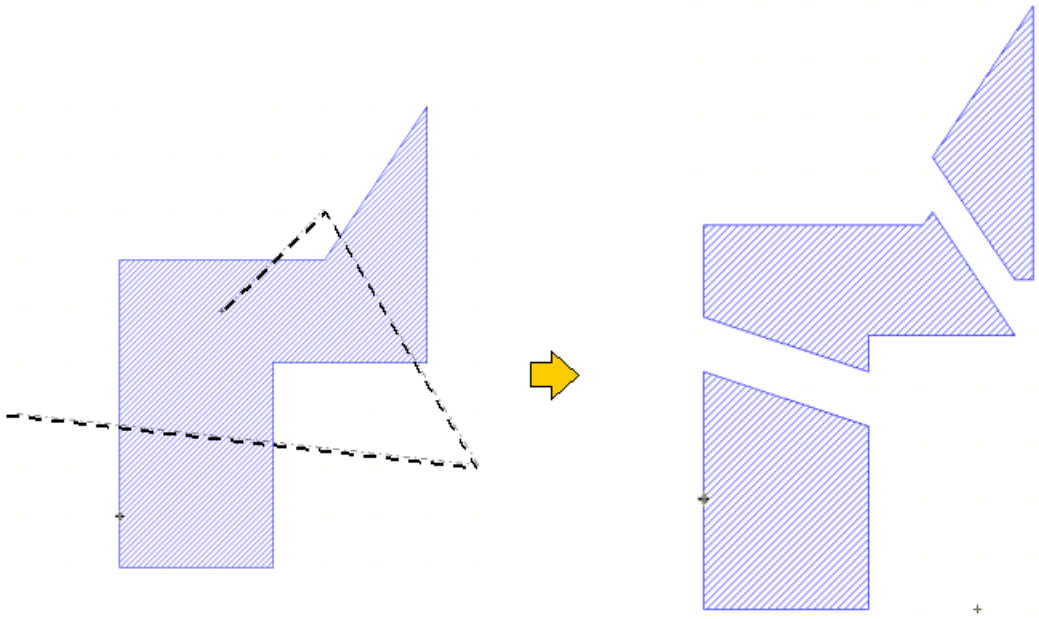


Figure 12-28: Cut by Line

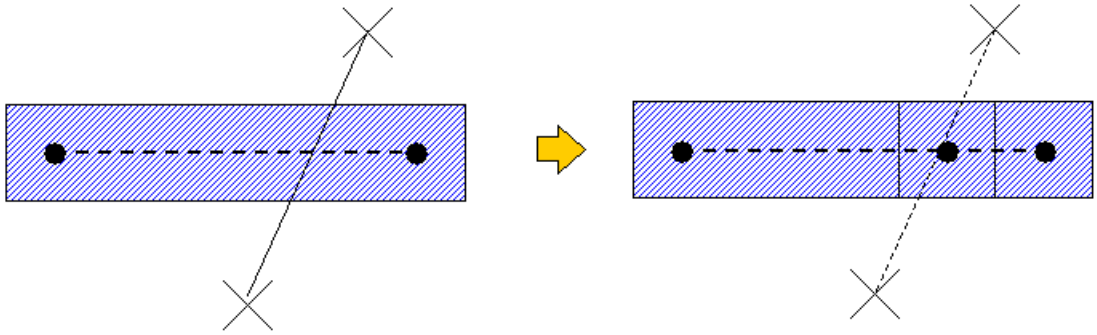


Figure 12-29: Cut Wires by Line

12.8: Cut by Vertices Count

This section describes the **Tools→Cut by Vertices Count** submenu commands. The commands cut large regions (by means of vertices count), and/or wires into smaller parts with a specified number of vertices. It simplifies editing layout objects, and optimizes the storing and handling of layout objects by the EXPERT database. Cutting objects must comply with GDSII restrictions on vertices count for objects of wire and region types.

You can choose the following sets of objects for a cut operation:

- All selected wires and regions in the active cell view;
- All wires in the current project;
- All regions in the current project;

- All regions and wires in the current project.

The **Tools→Cut by Vertices Count→Cut by Vertices Count Setup** commands pops upa dialog for setting the maximum number of vertices used by the **Cut** command for resultant split wires and regions (see Figure 12-30).

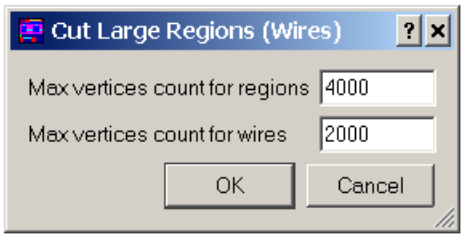


Figure 12-30: Cut Large Polygon (Wires)

12.9: Split Compound Objects

The **Split** commands allows you to split arrays, parametric wires, and multipaths into separate objects: wires, boxes and instances. Once compound objects are split, you cannot modify them like a complex object. In addition, splitting compound objects can be performed during object creation (see Chapter 5: “Creating Layout Objects”).

The following object sets can be split:

- Selected arrays, P-wires, and mulipaths in the active cell view;
- All arrays in the current project;
- All P-wires in the current project;
- All multipaths in the current project;

The array splitting depends on array splitting mode selectable from the **Tools→Split→Array Split Mode** submenu: split all, split by rows, split by columns.

12.10: †Clip Out/Clip In

Purpose

To cut a rectangular area from the active cell layout and save it as a separate cell.

This operation preserves cell hierarchy whenever possible, i.e., for instances that fall completely within the clipped box. Object/layer visibility settings are ignored: Clipping is performed over all geometry.

Execution

Select the menu command, **Tools→Clip Out**. The cursor assumes the **ClipOut** shape, seen in Figure 12-31. Select the rectangular area to be clipped out by clicking at any two of its opposite corners.

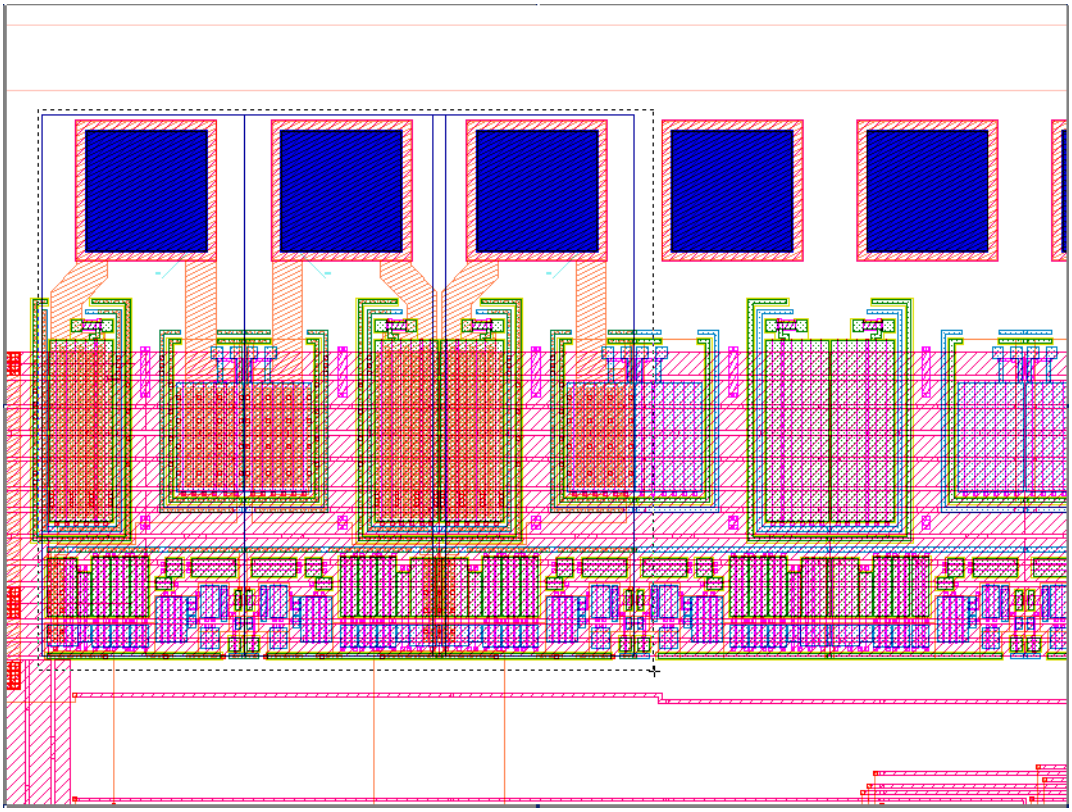


Figure 12-31: Clipping an Area Out

After the clip out box is selected, the **Cell Name Selection** dialog box appears with the caption, "Clipped cell name", prompting you to select the cell name and to store the clipped piece of layout. This dialog proposes a unique default name for the cell. You can replace it.

- Pressing **Cancel** stops the current clip out operation. You can then select another clip out box or other editing operations.
- If you press **OK** and you select an existing cell name, a warning box will appear. You must then press **OK** and repeat name selection.
- Otherwise, (i.e., the cell name was new), the clipping process begins. It may be that the clipping box contains no objects. The corresponding warning appears and the clipping box selection may be repeated.

Results

In Figure 12-32, an instance of `NewCell3` with some surrounding is cut from `TopCell`.

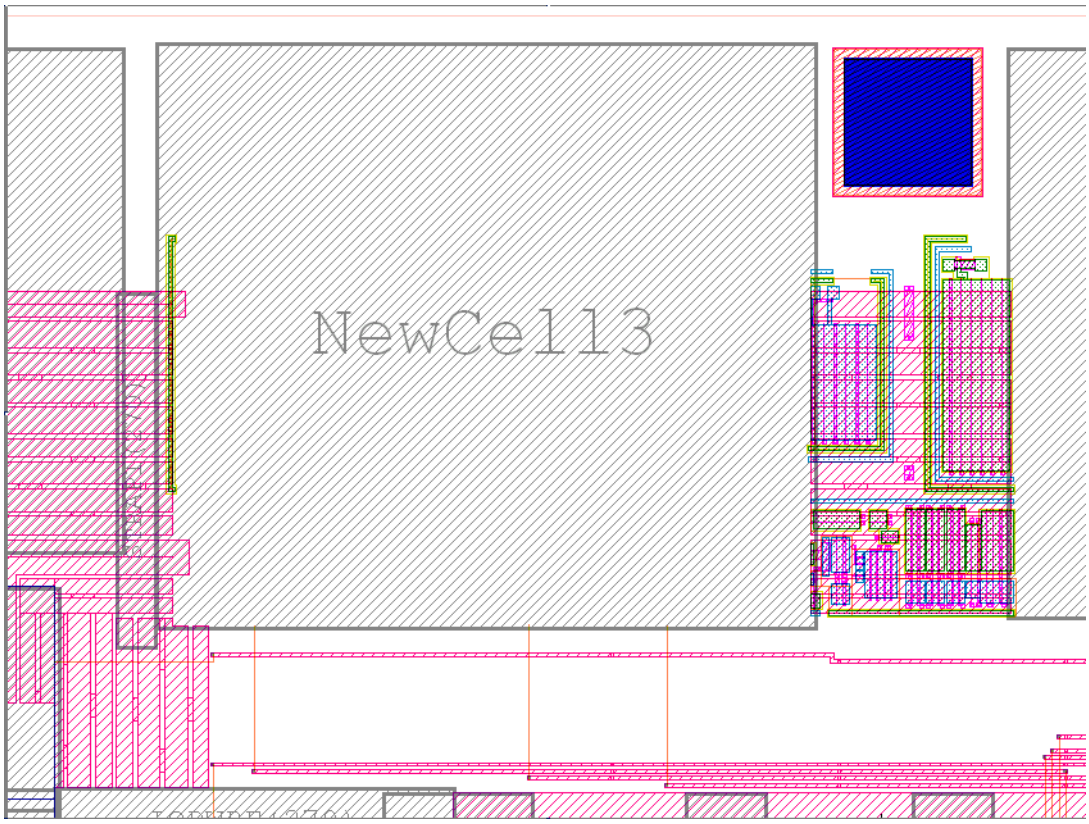


Figure 12-32: Results of Clip In

All objects (regions, wires, instances, etc.) that fall completely into the clip box are put into the new cell just as they are.

- If some instances from an array fall completely into the clip box, they constitute a subarray, and it is put into the new cell.
- For regions, wires and boxes that are cut by the clip box, only pieces that fall into the clip box are converted into regions and put into the new cell.
- If an instance or an array entry is split by the clip box, it is exploded and its elements are clipped out recursively.

A **Clip-in** operation is similar to a **Clip-out** operation. It creates a new cell from all geometry selected by the clip box, preserving cell hierarchy in the clipped piece of layout whenever possible. In addition, it replaces this clipped piece in the original cell by an instance of the newly created cell.

The relationship of Clip-in to Clip-out is similar to that of “Create Cell in Place” to “Create Cell” (from the selected objects).

12.11: †Merge Selected

Purpose

To perform a union of primitive objects that are selected in the active cell.

The objects processed by this operation will be replaced by nonintersecting regions whose area equals the union of the original objects. The operation does not change the contents of the EXPERT clipboard. The action may be invoked from the **Merge Selected** command of the **Tools** submenu.

All of merged objects, including wires, rectangles and circles, are converted to polygons.

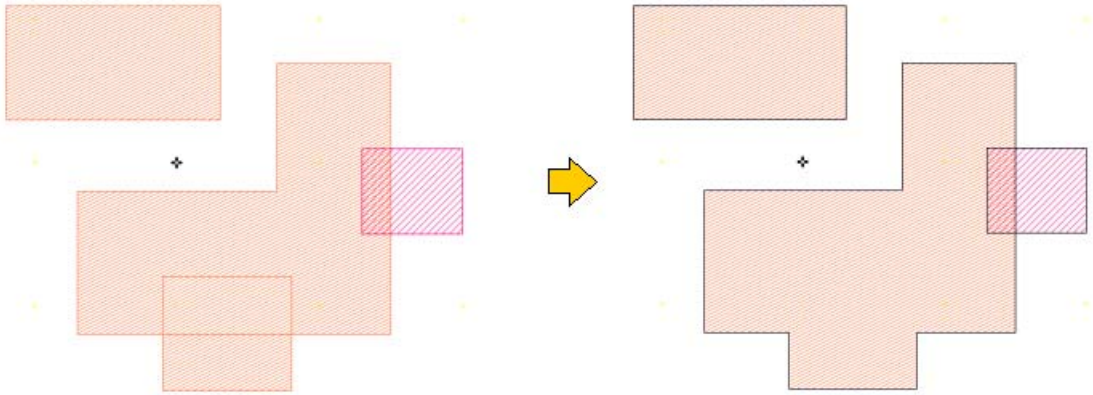


Figure 12-33: Merge Selected

Figure 12-33 illustrates how the layout looks before and after the **Merge Selected** operation. The left picture contains three selected (solid) objects: two boxes and one region. The upper left box is not intersecting with other selected objects so it is not modified. The right box (dashed lines) is not selected so it is not modified either. The lower box and region are replaced by a new region covering their merged area.

12.12: †Resize Selected

Purpose

Expands or shrinks areas bounded by selected objects of the current layer of the current cell. The operation shifts object boundaries inside (shrink) or outside (expand) the interior. The operation does not affect instances of cells, and it does not change the contents of the EXPERT clipboard.

All of merged objects, including wires, rectangles and circles, are converted to polygons.

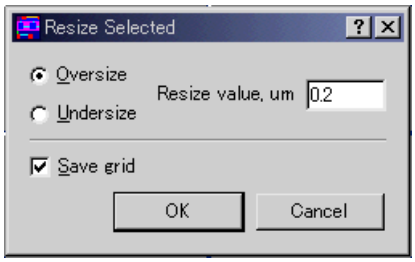


Figure 12-34: Resize Selected

Execution

If the **Tools→Resize Selected** command is invoked, the dialog shown in Figure 12-34 appears, allowing you to input the following parameters:

- **Type of operation** defines shifting direction: **Oversize** (expands the covered areas) or **Undersize** (shrinks the covered areas);
- **Value** defines the shift size; it is a positive number expressed in current measurement units, shown in the **Metric** bar;
- **Save grid** acts for region vertices whose adjacent sizes have slopes in 45 degree multiples. It states that resizing must be performed in such a way that the region vertices will remain on grid, if they were on grid and the resize value is a multiple of the grid size (possibly, unknown).
- **Run** button starts the **Resize** operation;
- **Cancel** closes the **Resize** dialog without modification of layout.

A reasonable **Value** parameter is required, otherwise **Oversize** and **Undersize** operation scan cause unexpected results. Figure 12-35 demonstrates several transformations of layout objects made using the OverSize operation. Note, the hole inside the first object is decreased, the hole inside the second object is eliminated by merged internal boundaries, and merged boundaries of the third object have created a hole.

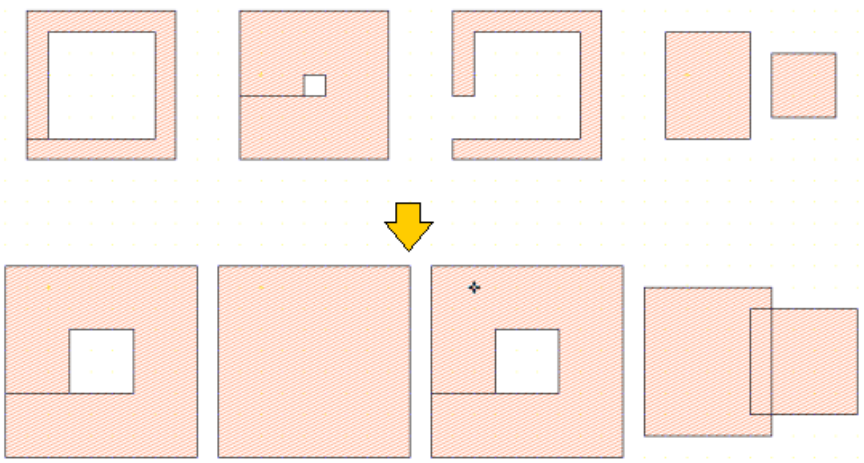


Figure 12-35: Resize Operations (Oversize)

Figure 12-36 demonstrates transformations made by the **Undersize** operation. Note, the first and the second objects are divided into two pieces. The hole inside the third rectangle is increased. The hole inside the fourth has become more than the object itself, so the object has disappeared.

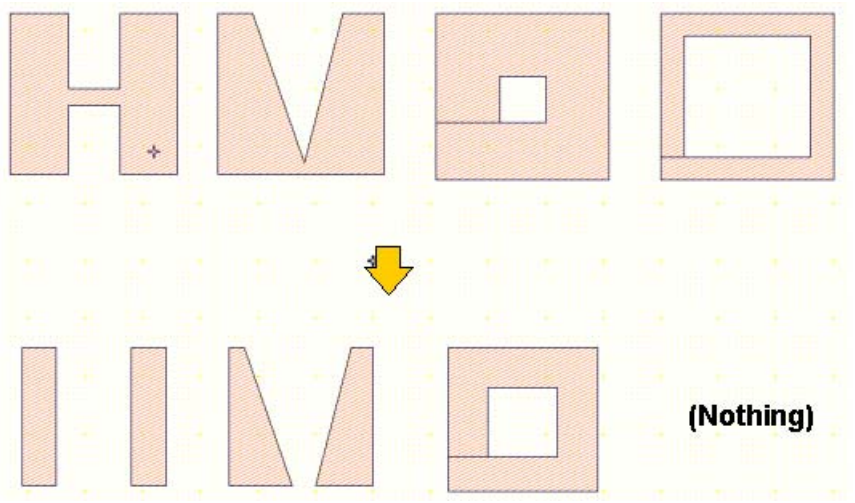


Figure 12-36: Resize Operation (Undersize)

12.13: Technology Migration

12.13.1: †Scale Selected and Scale Whole Layout

The set of selected objects or the whole layout of the edited cell is rescaled according to a number of parameters specifying the scaling factor (amount of scaling), center, and options of scaling as shown in Figure 12-37.

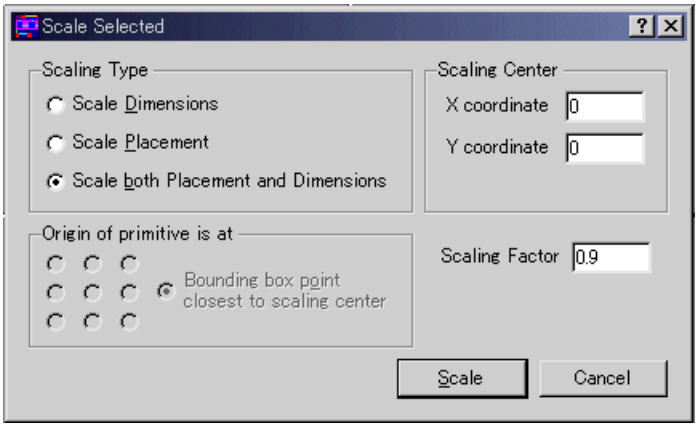


Figure 12-37: Parameters of Scaling Operations

The meaning of most parameters is evident. If **Scale Dimensions** or **Scale Placement** types are selected, it is necessary to specify individual scaling reference points (origins) for all objects that undergo scaling. The scaling parameter panel provides some natural choices. The nine radio buttons in a 3x3 pattern show possible choices with respect to the bounding box of the object. For example, in Figure 12-37, the scaling center is set to be the upper left corner of the bounding box of the object(s).

Hint: It is possible to set the scaling center by mouse: If you set a temporary reference point (default shortcut "Q"), and leave the scaling center at (0,0), then scaling will be performed with respect to the temporary reference point.

12.13.2: †Snap Selected (or Whole Layout) to Grid

The set of selected objects (or the whole layout of the edited cell, respectively) is snapped to the specified grid according to parameters shown in Figure 12-38.

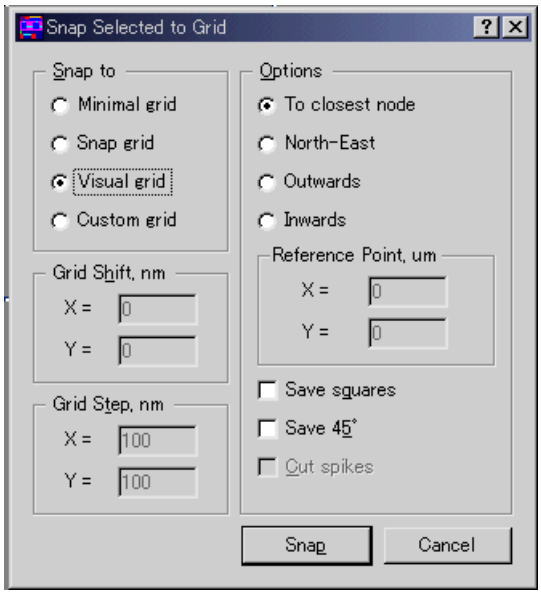


Figure 12-38: Parameters of Snapping to Grid

The first group of parameters specifies the grid. If the option, **Custom grid** is selected, then it is possible to specify both grid steps in X and Y direction and the displacement (shift) of the grid with respect to the origin.

The second group of parameters specifies how the coordination of the off-grid points will be modified. These parameters may help to preserve symmetries, distances, and other shape properties.

To closest node: The off-grid point will be shifted to the closest grid point.

North-East: The off-grid point will be shifted to the closest grid point so that both coordinates increase.

Outwards/Inwards: The off-grid point will be shifted to the closest grid point in the direction outwards or inwards with respect to the specified reference point.

Save squares: In this mode, the sides of square boxes will be kept equal.

Save 45°: Angles in multiples of up to 45 degrees will be preserved.

To save a 45° geometry, there is an option **Cut spikes**. If this option is on, sometimes 45°corners are cut to a small extent to place the snap on grid with minimal change of area. If this option is off, one of the adjacent sides is shifted.

Hint: It is possible to set the reference point by mouse: If you set a temporary reference point (default shortcut "Q"), and leave the scaling center to be (0,0), then scaling will be performed with respect to the temporary reference point.

12.14: Handling Derived Layers

Normally, a derived layer is recalculated when you activate it (by clicking on its color strip in the Layer bar).

The following commands from the **Tools→Derived Layers** submenu handle recalculation of derived layers.

Freeze Derived Layers blocks their recalculation during their activation.

Generate Active Derived Layer invokes generation of the active derived layer.

Generate All Derived Layers invokes generation of all derived layers. This command cleverly rebuilds all derived layers: The layers are generated according to the logical order of their derivation.

Clean Derived Layers: This submenu has commands to remove all geometry from derived layers in the top level of the current cell, in the whole hierarchy of the current cell or in the whole current project.

Save Full Derivation Script: This command allows you to store the DRC script that generates all derived layers according to the current technology, into a `dsf` file. The file name is selectable by the file browser. The result of the execution of this script is the same as for **Tools→Derived Layers→Rebuild All** command with derivation settings same as in the DRC setup.

Derivation Bar: This command shows/hides the **Layer Derivation** bar described in Figure .

Often running DRC scripts produces many new layers, and afterwards it is desirable to delete all geometry from these "insignificant" layers. One way is as follows:

- save layout before running DRC
- run DRC
- inspect report
- close project without saving
- reload the project

However these "extra" layers may be useful while inspecting and fixing DRC violations, and the "inspect report" step is in fact "inspect and fix", so quitting without saving would not be desirable.

Therefore you may do the following:

- Run your DRC script
- Using the **Setup→Technology→Layer/Wire Setup** menu command, set all these new layers to be derived, with dummy derivation statement that consists of a single semicolon (`;`) character (end of statement marker).

12.15: Merge Two Wires

With this tool you may merge two wires from the same layer.

You must select a couple of wires in advance. When you execute this command, the nearest vertices of those wires are connected.

If the wires have different parameters (width, end/join style), then the parameters of the first wire will be used for the resultant wire.

This command can work also for separated wires, but in this case, it may generate unexpected results shown as Figure 12-39.

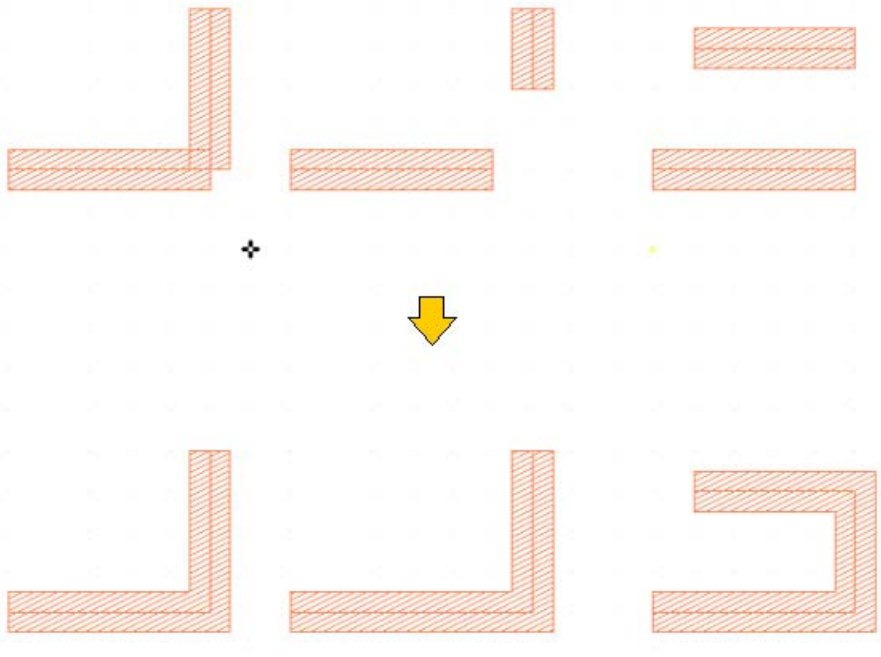


Figure 12-39: Merge Two Wires

12.16: Various

12.16.1: Scratch Layers

The **Tools→Scrath Layers** submenu contains two commands:

- **Clean Scratch Layers:** Delete cell primitives, cell hierarchy or delete data from all scratch layers in whole project.
- D**elete Scratch Layers from Technology**

12.16.2: Pick Layer

If you select **Tools→Pick Layer** and click at a shape or text, then the layer of this object becomes active (so that you can draw in this layer or copy selected objects into it). Preselection helps you to pick the required object.

12.16.3: Polygonize

The commands of the **Tools→Polygonize** submenu convert wires and circles/ellipses to polygonal regions. It is possible to:

- selected objects,
- all wires in the project,
- all ellipses/circles in the project.

It also converts selected texts to polygonal ones.

12.16.4: Mouse Click

It is possible to assign bindkeys to left, right and idle button mouse clicks using the commands from the **Tools→Mouse Click** submenu. This option is useful, e.g., if your mouse is unstable or you are using a tablet as a pointing device with a stylus that has some buttons inconvenient to click.

12.16.5: Attach Texts

The **Tools→Attach Texts** command attaches a selected text to another selected object. The layer of the attached text becomes same as the parent object.

12.16.6: Detach Text

The **Tools→Detach Text command** is a convenient operation to detach all attached texts from all selected objects.

12.16.7: Previous Tool

The **Tools→Previous Tool** command switches to the previously used drawing tool.

12.16.8: Void Tool

The **Tools→Void Tool** command switches to the Void tool. This tool does nothing and is used for safety while idly browsing the layout.

Chapter 13:
Parameterized Cells

13.1: Overview

Parameterized cell (P-cell) is a cell generator programmed with Xi scripting language. Each P-cell has predefined set of parameters. P-cells are stored in the EXPERT database in the form of Xi code. Using P-cells in the EXPERT environment is extremely easy. When you place instance of a P-cell, the Numeric Input form shows the list of parameters for a specified P-cell (see Figure 5-9). You can accept defaults or change value of any parameter. Using P-cells gives a considerable reduction of disk space in the case of multiple instancing of a cell with different parameter values. EXPERT creates a temporary regular cell for each set of parameter values, and does not store it in the database.

There are two types of P-cells in the EXPERT design environment: graphical and textual ones.

Graphical P-cell is a simple P-cell generated automatically by specifying expressions for the predefined rules, such as stretch, repetition, layer, and so forth. Their functionality is limited by supported parameters. Graphical P-cell is suitable for quick and intuitive creation of simple parameterized cells. Textual P-cell is a complex P-cell with structures described using powerful features of LISA/Xi scripting language that are impossible to generate automatically with a graphical P-cell.

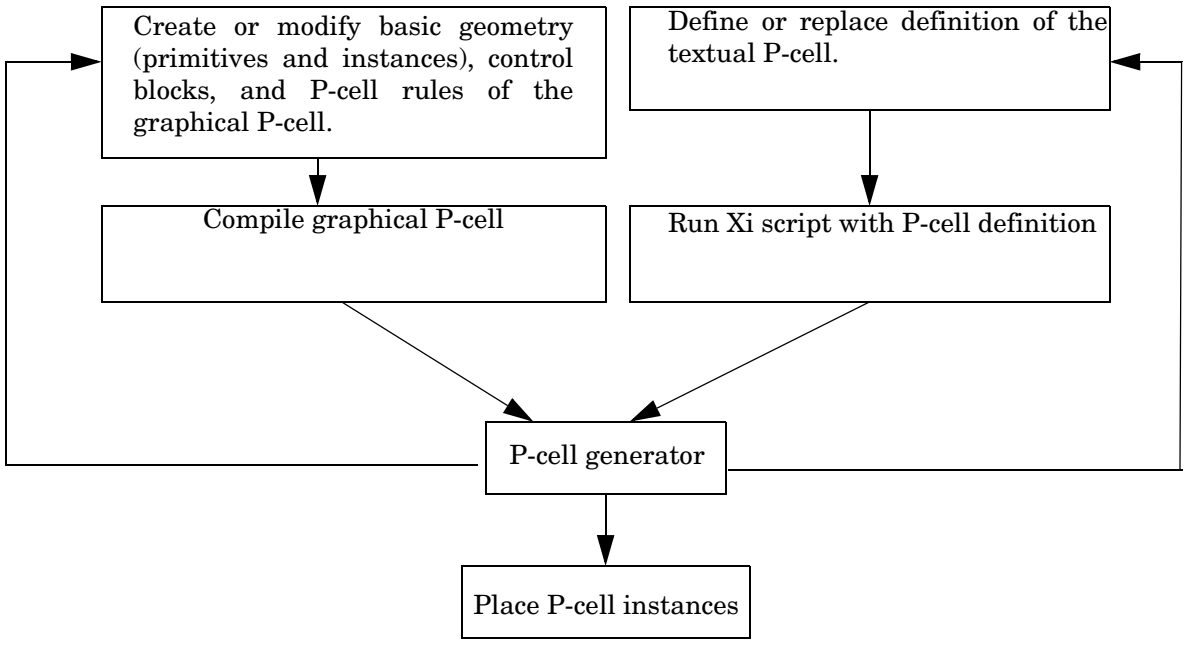

Create or modify basic geometry (primitives and instances), control blocks, and P-cell rules of the graphical P-cell.
Define or replace definition of the textual P-cell.
Compile graphical P-cell
Run Xi script with P-cell definition
P-cell generator
Place P-cell instances

Figure 13-1: P-cell Design Flow

13.2: P-cell Menu Command

The Pcell menu command (see Figure) is to create and modify p-cells. The menu includes the following commands (see Figure 4-1):

New: Creates new graphical or textual P-cell.

Parameters: Defines parameters for the graphical P-cell.

Preface: Defines user defined Xi script code to be added to automatically generated graphical P-cell code.

Create Control Block: Creates a graphical P-cell control blocks.

Compile: Creates or modifies a P-cell in the EXPERT database from the current graphical P-cell.

Show Xi Script: Show P-cell script.

Recalculate Instances: Recalculates all generated cells.

View Pcells: Shows all generated cells.

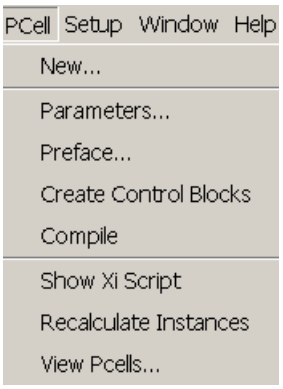


Figure 13-2: P-cell Menu

13.3: Creating graphical P-cells

13.3.1: Graphical P-cell Creation Flow

- Create a new graphical P-cell using the command **Pcell→New** (see Figure 13-3) with the option **Graphical Pcell** checked. The initial content of the new graphical P-cell can be filled with content of a regular cell by checking the option **Compose of selected cell** and selecting the cell name in the **Cell List** control of the New PCell dialog form.

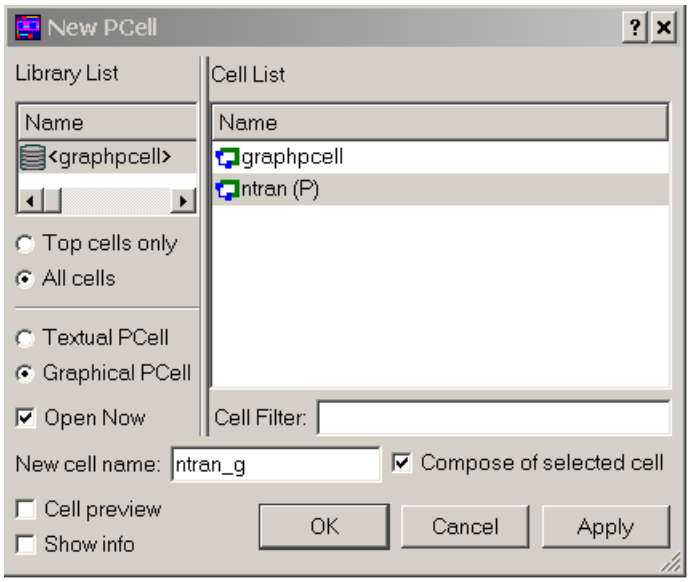


Figure 13-3: New P-cell Dialog

- Enter a basic design (primitives and instances) using normal editing commands. If you have one in an already existing cell, put the instance and explode, or copy and paste those primitive objects using the commands in the **Edit→Expert Clipboard** menu option.
- Create P-cell control blocks using **Pcell→Create Control Block** command. A graphical P-cell control block is a special object for storing common graphical P-cell rules for all objects inside or crossed by a control block. The process of creation and modification of a P-cell control blocks is similar to polygonal region objects. As you create a control block, the **PCell Rules** dialog panel pop ups (see Figure 13-4). The **PCell Rules** dialog allows you to add various graphical p-cell rules used during auto generation of a P-cell from a graphical one.

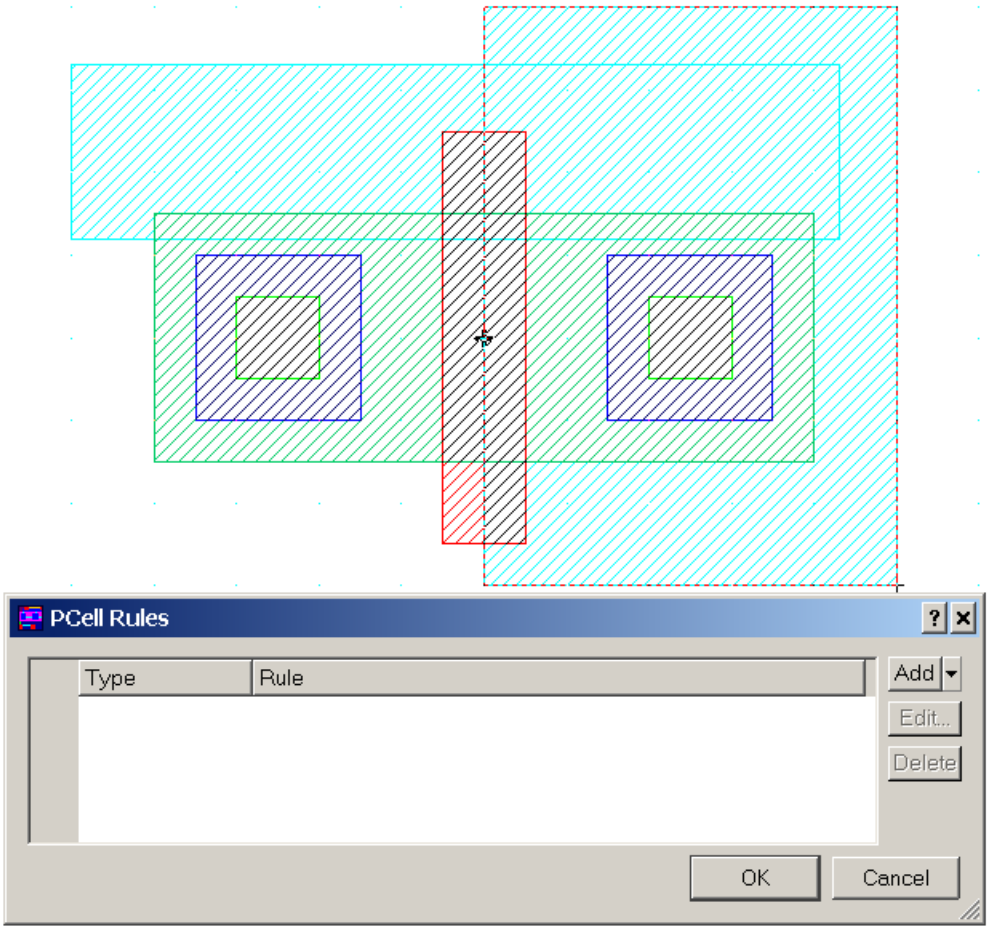


Figure 13-4: P-cell Control Block with PCell Rules Dialog

- Add graphical P-cell rules definitions to the control blocks, or (and) objects of the P-cell by selecting appropriate objects and editing the **Pcell Rules** column in the **Expert Property** bar by pressing **...** button (see Figure 13-5). Several rules can be applied to one object/control block, as well as one rule can be added to several object/control blocks.

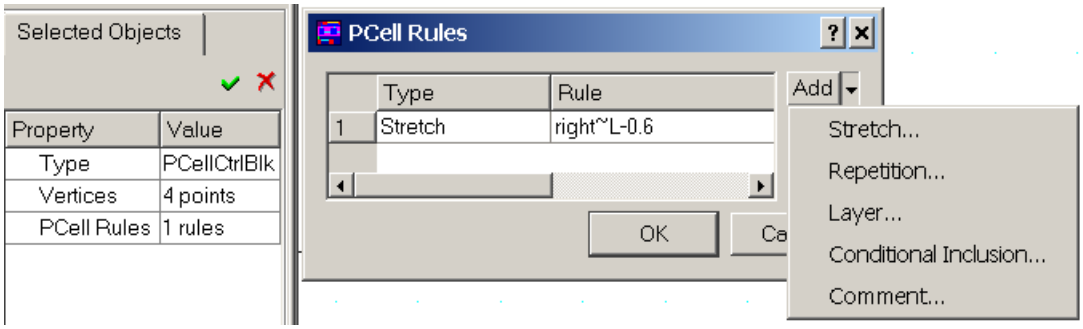


Figure 13-5: Pcell Rules Dialog

- Add P-cell parameters used when you place instance of P-cell and define default values for them using the **PCell→Parameters** command.
- Compile graphical P-cell (**PCell→Compile**) to produce P-cell.

- Test a graphical P-cell by placing P-cell instances with different parameter values, or using the **PCell→PCell View** command. The last one shows a dialog form with a list of calculated cell a corresponding to a specific set of parameter values (see Figure 13-6). You can add a new set of parameter values, or remove the selected calculated cell if the last one is not instanced.

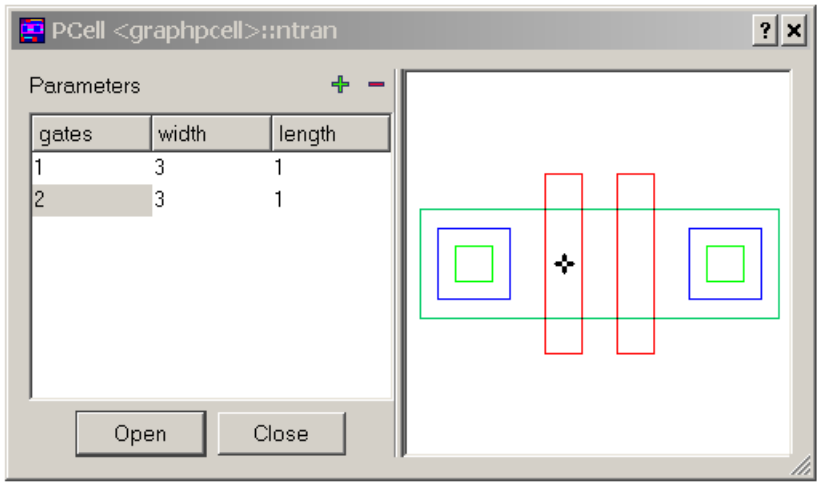


Figure 13-6: PCell Viewer

13.3.2: Defining Graphical P-cell Rules

Automatic generation of a P-cell from graphical P-cell is performed on the basis of the predefined P-cell rules. P-cell rules can be assigned both for a P-cell control block, and for primitive or instance objects. If a P-cell rule is defined for a control block, then all layout objects that are included in or intersected by the control block will be affected by a specified P-cell rule. The following types of predefined P-cell rules can be specified for graphical P-cell objects in **PCell Rules** dialog panel:

- **Stretch** defines a p-cell rule for stretching objects. Use the Add?Stretch button in PCell Rule Dialog to activate Stretch form (see Figure).

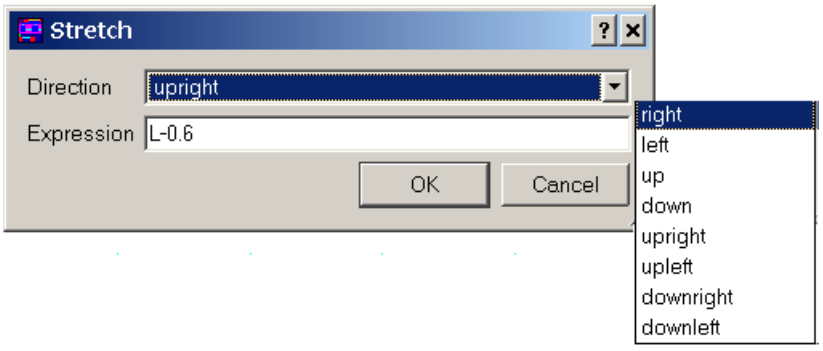


Figure 13-7: Stretch Dialog

Stretch rule changes the size and (or) position of layout objects. Stretch direction is defined by the **Direction** list box control. The following stretch directions can be assigned: right, left, up, down, up-right, up-left, down-right, down-left. The value of the stretch can be defined by any valid LISA/Xi expression. Objects that are intersected by P-cell control blocks are stretched, and objects with their own stretch rules, or objects included in control blocks, are moved in the specified direction. If the value of the stretch is negative, then the stretch is performed in the opposite direction.

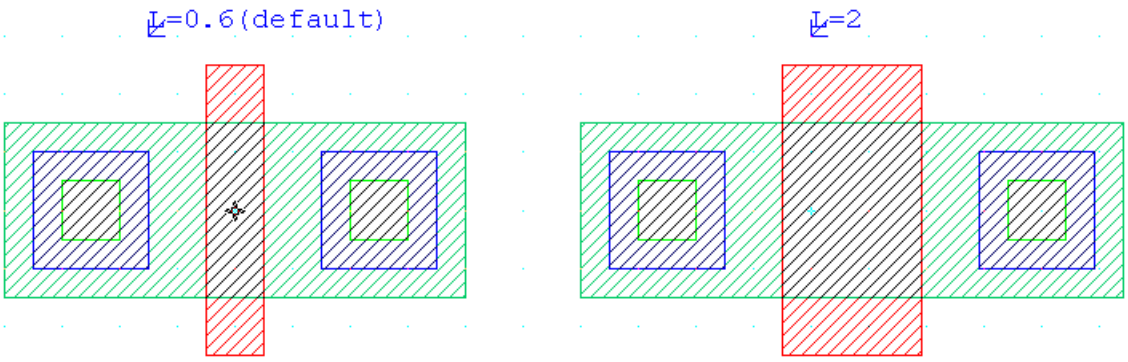


Figure 13-8: Stretch Example (Direction:right; Expression:L-0.6)

- **Repetition** rule are to repeat objects in the specified directions. Use the **Add→Repetition** button in **PCell Rule** dialog to activate the **Repetition** form (see Figure 13-9).

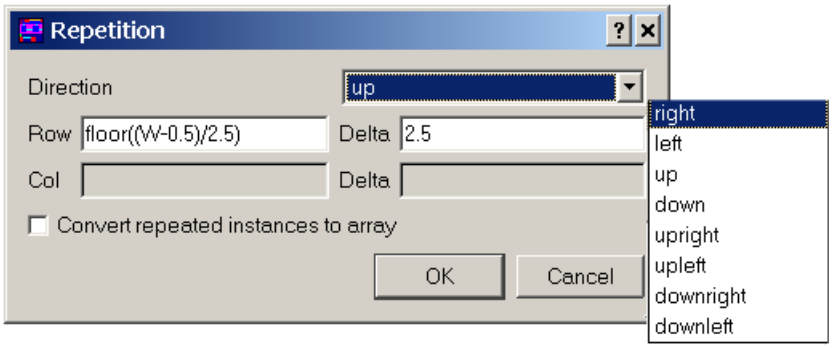


Figure 13-9: Repetition Dialog

The following repetition directions can be defined: right, left, up, down, up-right, up-left, down-right, down-left. If you choose the up-right, up-left, down-right, down-left directions, the target objects will be created in the form of array (see Figure 13-10). The number of repetitions is defined in the **Row** and **Col** editing fields by any valid LISA/Xi expression. A LISA function floor( ) is useful to truncate the calculated value to the integer number. The repetition step in vertical (row) and horizontal (column) directions is defined by the corresponding **Delta** editing fields. If **Convert repeated instances to array** is checked, then instance objects with the Repetition rule will be converted to instance arrays.

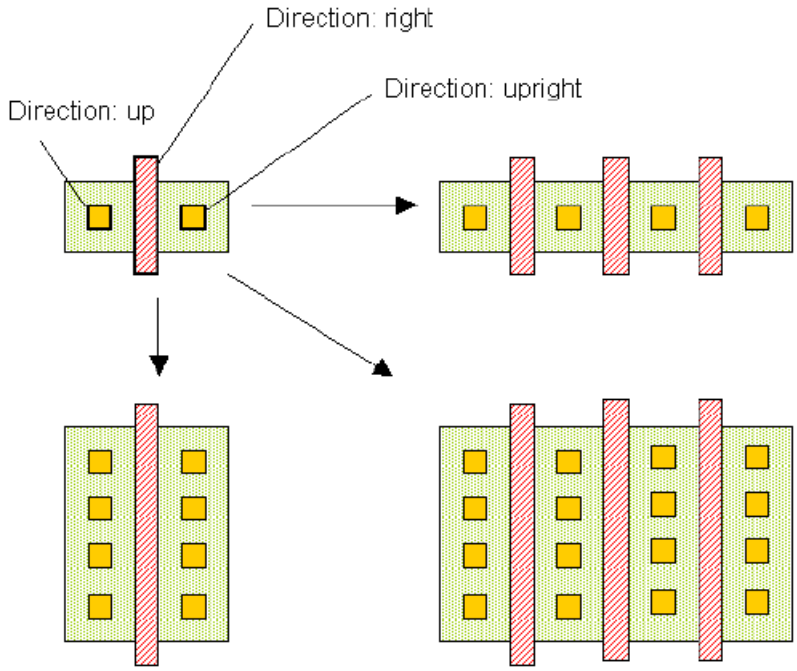


Figure 13-10: Repetition in Various Directions

- **Layer** rule is to parameterize a layer for primitive objects. Use the **Add→Layer** menu option in the **PCell Rule** dialog to activate Layer form (see Figure 13-11). The value for Layer rule can be any valid LISA/Xi expression evaluated as string type, and matched with a layer name defined in the technology.

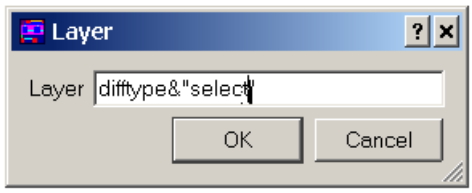


Figure 13-11: Layer Dialog

- **Conditional Inclusion** rule allows you to include (exclude) objects or a group of objects defined by the control block from the calculated cell, depending on how the specified condition expression is evaluated (true or false) and which option (**Show** or **Hide**) is checked. Use the **Add→Conditional Inclusion** menu option in the **PCell Rule** dialog to activate the **Conditional Inclusion** form (see Figure).

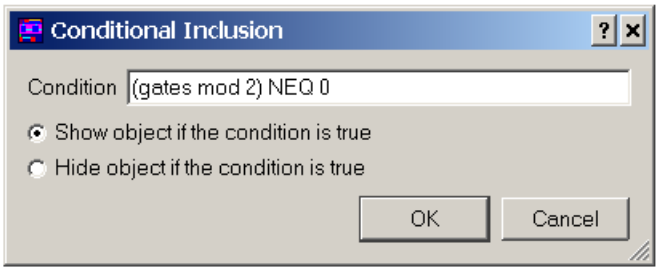


Figure 13-12: Conditional Inclusion Dialog

- **Comment** rule allows you to add a specified comment line to the group of objects defined by the control block when P-cell script is compiled from the graphical P-cell. It will allow a more convenient editing of the P-cell script if the current graphical P-cell is converted to a textual one. Use the **Add→Comment** menu option in the **PCell Rule** dialog to activate the **Comment** form (see Figure 13-13).

Figure 13-13: Comment Dialog

- **Instance Parameters** rule allows the assigning of parameters for child P-cell instances in the current graphical P-cell. This P-cell rule can be applied only to P-cell instance objects. Additional command **Instance Parameters** appear when you press the **Add** button in **PCell Rules** form, when P-cell instance is selected. An **Instance Parameters** form (see Figure 13-14) contains a list of names of all parameters of the instanced P-cell and an Expression field for each parameter.

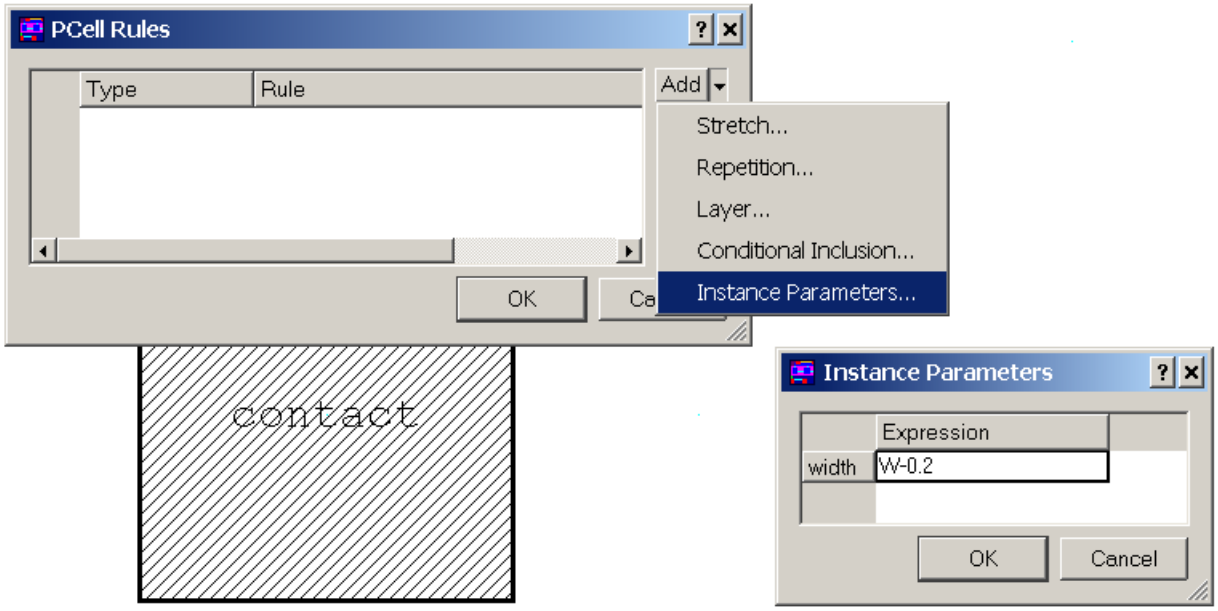


Figure 13-14: Instance Parameters Dialog

13.3.3: Defining Graphical P-cell Parameters

The command **PCell→Parameters** shows the **PCell Parameters** dialog (see Figure 13-15). There are three fields for each parameter: name, data type and default value. Parameter name is shown during instantiation of P-cell in the Numerical input form. Parameters names can be used in the graphical P-cell rules also. The following LISA/Xi types are valid for P-cell parameters: Double, Integer, String and Boolean. Default parameter values are used in the Numerical Input form as initial ones.

Press the + button to add a new parameter. The − button deletes the selected parameter. Use ↑ and ↓ buttons to change parameters order.

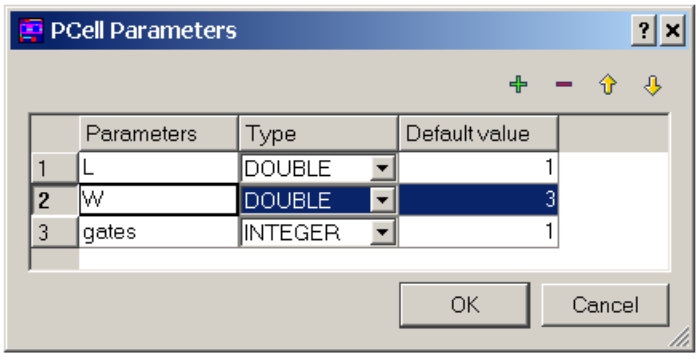


Figure 13-15: PCell Parameters Dialog

13.3.4: Graphical P-cell Preface

You can add some user defined LISA/Xi code to the graphical P-cell using the command **PCell→Preface**. For example, you can define and initialize some internal variables for using in the P-cell rules. This code will be added to the automatically generated P-cell code at the beginning.

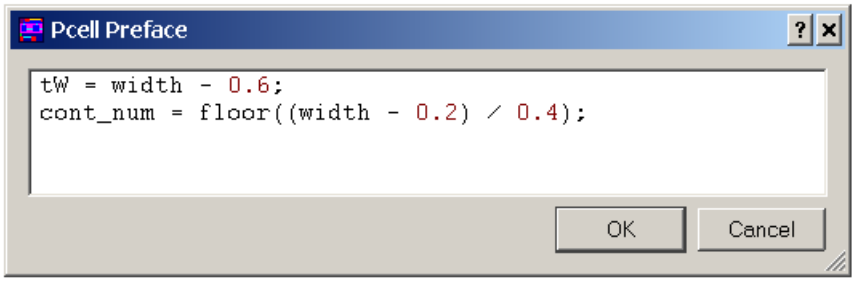


Figure 13-16: PCell Preface Dialog

13.3.5: Graphical P-cell Compiling

The **PCell→Compile** command applies changes made by editing of graphical objects and control blocks, P-cell rules, and preface script to the P-cell stored in the EXPERT database. Check warning and error messages are displayed in the **Expert Output** window to ensure that P-cell generation was successful. Check cell generated by P-cell with various sets of parameters is in the **PCell Viewer** window (command **PCell→View Pcells**). To apply changes made to P-cell for all instanced P-cell variations, run the **PCell→Recalculate Instances** command.

13.4: Creating Textual P-cells

13.4.1: Textual P-cell Creation Flow

- Create a new textual P-cell using the command **Pcell→New** (see Figure 13-17) with the option **Textual Pcell** checked. An initial script of the new textual P-cell can be exported from content of a regular cell by checking the option **Compose of selected cell** and selecting cell name in the **Cell List** control of the **New Pcell** dialog form.
- After specifying the cell name and pressing the **OK** button, the **PCell Parameters** dialog pop ups, allowing you to define the P-cell parameters. However, you can add new or modify existing parameters at any time, so you can click **OK** with no parameter definition. The Xi-script editor will be activated with a predefined skeleton for P-cell definition.
- Edit textual P-cell parameters and body Xi-script manually.
- Run Xi-script to create new or modify existing P-cell.
- Test cells generated by P-cell in the in the **PCell Viewer** window (command **PCell→View Pcells**). To apply changes made to the P-cell for all instanced P-cell variations, run the **PCell→Recalculate Instances** command.

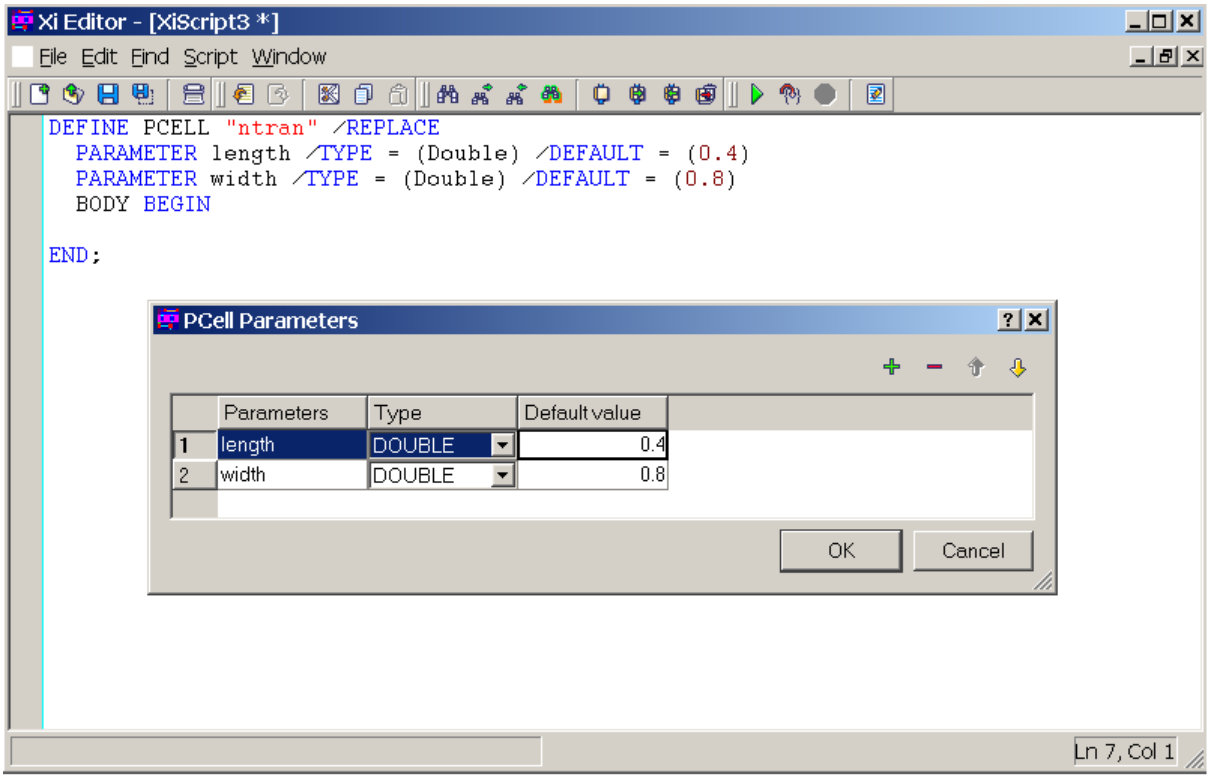


Figure 13-17: Textual P-cell Creation

You can create textual P-cells from scratch in the Xi-script editor using the predefined Xi command `DEFINE PCELL`.

13.5: Conversion From Graphical to Textual P-cell

The **PCell→Show Xi Script** command is displayed as a Xi script for both graphical and textual P-cells. You can edit and apply changes only to textual P-cells with the **Script→Run** command from the Xi Script Editor. When you try run a script with a P-cell definition, and there is a graphical P-cell with the same name in the current database, a warning message will pop up (see Figure 13-18).

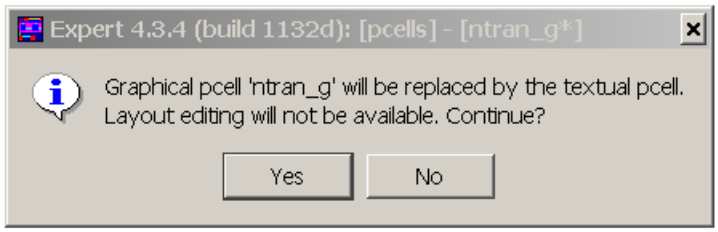

Expert 4.3.4 (build 1132d): [pcells] - [ntran_g*]
Graphical pcell 'ntran_g' will be replaced by the textual pcell.
Layout editing will not be available. Continue?
Yes
No

Figure 13-18: Graphical P-cell Conversion Warning

Once graphical P-cell is converted to a textual one, you will not be able edit the last one in the graphical way. A conversion from textual to graphical format is not supported.

13.6: P-cell Password Protection

EXPERT can prevent unauthorized users from viewing and editing P-cell script code. To protect P-cell in the current project, check the **Use Pcell password protection** option on **Data Safety** page of **Editor/Viewer** setup section of the **Expert Preferences** dialog (**Setup→Editor/Viewer**). The **Pcell Password Protection** (see Figure 13-19) allows you to enter a password. A password is case-sensitive, can be cup to 128 characters long, and can contain any combination of letters, numbers, and symbols. As you type the password, the * symbol is displayed for each character you type.

Figure 13-19: Pcell Password Protection Dialog

To clear or change an existing P-cell password, click **Change Password** on the **Data Safety** page. You will be asked for an existing and a new password (see Figure 13-20). Leave the **New Password** and **Confirm Password** fields empty if you would like to remove the P-cell password protection.

Figure 13-20: Resetting Pcell Password

To show and/or edit a password protected P-cell you need to provide correct password. The **Verify Pcell Password** dialog will be shown when you execute the **Pcell→Show Xi Script** command. If you failed to provide valid password, P-cell script will no be shown.

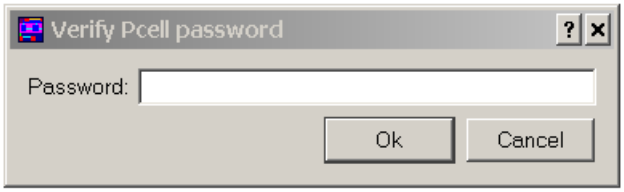


Figure 13-21: Verify Pcell Password Dialog

This page is intentionally left blank.

Chapter 14:
Integrated Layout Verification Tools

14.1: Layout Verification Tools

This chapter describes EXPERT's interface to layout verification tools (Figure 14-1) from Celebrity suite:

- GUARDIAN DRC
- GUARDIAN LPE Netlist Extractor (including parasitic RC extraction by HIPEX-RC)
- GUARDIAN LVS Verification

as well as

- Real-time DRC
- Electrical Node Probing (including Short-locator)

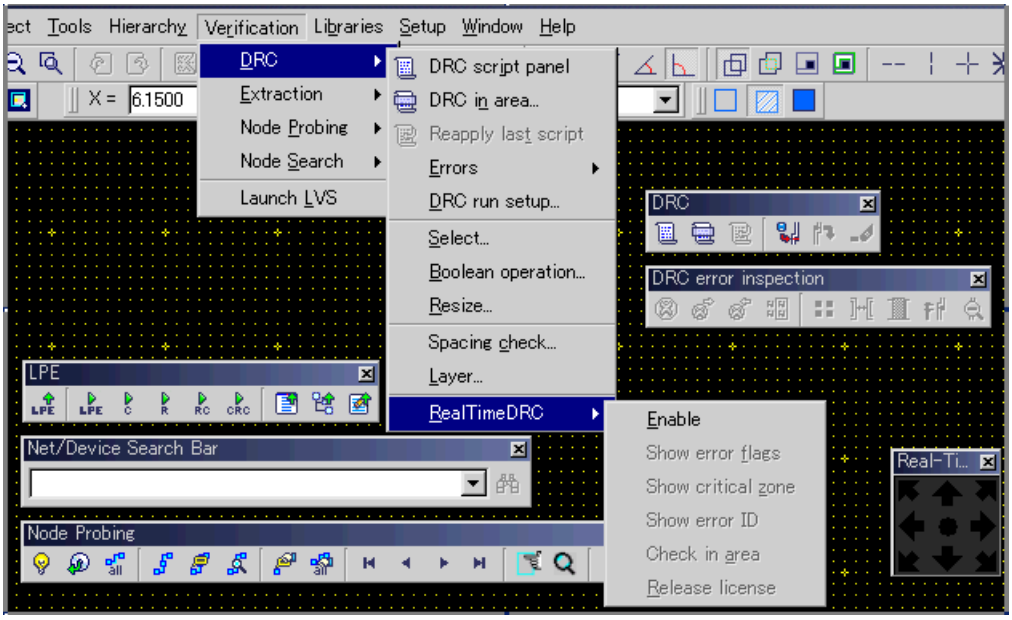


Figure 14-1: Elements of Interface to Layout Verification Tools in Expert

GUARDIAN DRC, LPE and LVS may run in three modes:

- as standalone applications,
- in batch mode, from the operation system's command line,
- integrated within EXPERT layout editor.

Complete descriptions of GUARDIAN DRC, LPE and LVS can be found in the GUARDIAN "LAYOUT VERIFICATION USER'S MANUAL".

14.2: Geometrical DRC System

14.2.1: Basics of Usage in Expert

GUARDIAN DRC geometrical DRC system integrated with EXPERT may be used in several ways:

- Run DRC scripts inline within EXPERT, immediately inspect DRC results and perform corrections.
- Run batch DRC jobs using xi-scripts to schedule runs of multiple DRC scripts on multiple layouts.
- Set Real-time DRC to run DRC checks in real time; design rule violations are reported immediately while the designer edits the layout.
- Use interactive DRC operations from the **Verification→DRC** submenu to run simple, one-command checks and to perform Boolean operations on layers.
- Run combined xi/DRC scripts and perform complicated operations with layout, such as area filling and layout comparison.

Interactive DRC commands may be invoked either from the **Verification→DRC** submenu or from the **DRC** bar.

DRC scripts in EXPERT can be run from the **DRC Script Panel**, opened by the **Verification→DRC→Script Panel** command or from xi-scripts.

Preparation of DRC scripts is facilitated by the following means:

- **Import** command at the DRC script panel, which allows you to import DRC decks from foreign formats.
- Several **Script auto-generation** commands in "Insert" menu of DRC Script Panel. They insert DRC statements into a DRC script basing on simple input forms (see Figure 14-3), so that the user doesn't have to worry about the exact syntax of commands.

Note: The complete description of GUARDIAN DRC can be found in the GUARDIAN LAYOUT VERIFICATION USER'S MANUAL.

DRC Script Running

To run a **DRC script**, perform the following steps:

1. Load IC layout.
2. Open the DRC Script Panel.
3. Load an existing EXPERT script using the **Load** button, or create a new script.
4. Choose the scope of layout objects which will be checked during the running of the DRC script by means of **Scope** group of radio buttons.
5. Press the **Run** button on the DRC Script Panel.

The way the DRC script runs depends on the settings assigned by the **Setup→DRC Run...** command, see Chapter 11: "Setup", Section 11.11: "DRC Script Run Setup".

After running the script, you can look at DRC results using the error report tools.

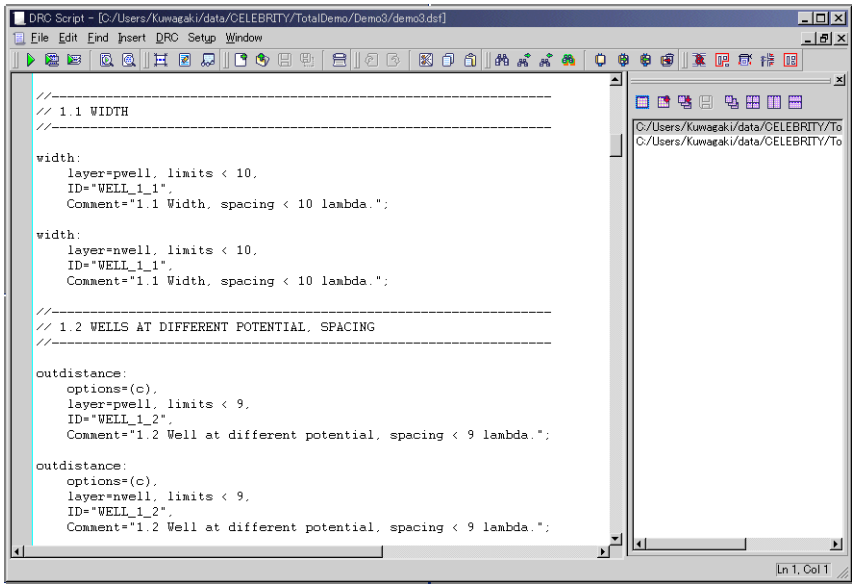


Figure 14-2: DRC Script Panel

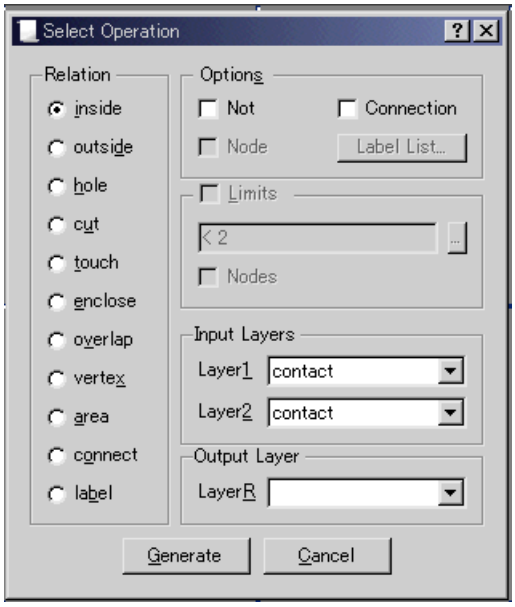


Figure 14-3: DRC Command Input Form Example: “Select” command

DRC Reporting

DRC results can be reported in three ways:

- as temporary output layers directly on the layout.
- as layers in error cells written into an external GDSII file.
- as GUARDIAN DRC error database, which is the most flexible and convenient way.

When errors are reported as regions to output layers, you may switch visibility of all irrelevant layers off and examine those input layers that are overlaid by output layers.

When errors are reported into the Error Data Base, you can inspect errors consecutively using the commands from **Verification→DRC→Errors** submenu or from the DRC toolbar.

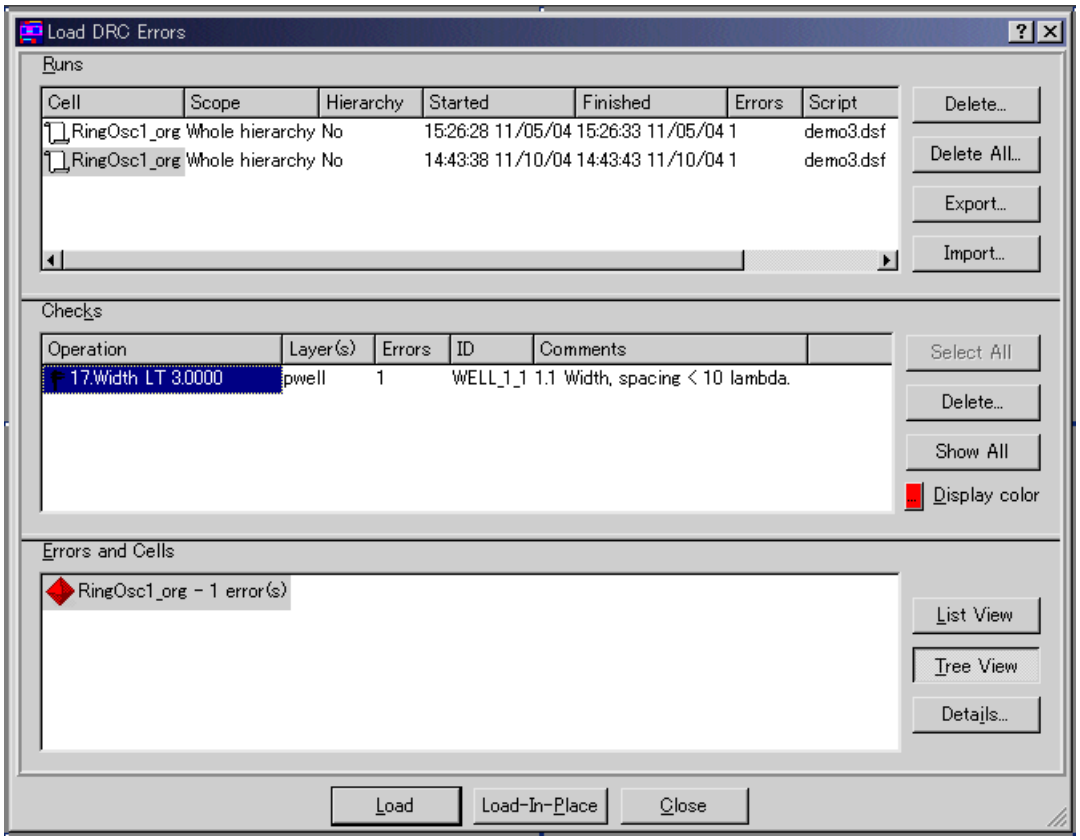


Figure 14-4: DRC Load Error Dialog

Reapply Last Script

Selecting **Verification→DRC→Reapply Last Script** allows you to start the last loaded script without entering in DRC Script Panel. This command is useful when you would like to start the last script without changing any settings for the current script.

DRC in Area

Selecting **Verification→DRC→DRC in Area...** allows you to run a DRC script for a specific part of an IC layout rather than the whole layout. After starting this command, you should mark a rectangular area on the IC layout area. The DRC Script Panel will then open, and you will have to load or create a DRC script and click the **Run** button. This script will be run only for the IC layout of the marked area.

14.2.2: Real-time DRC

Summary

Real-time DRC is a powerful feature of EXPERT that checks geometrical design rules during construction, in other words "on the fly". Rules for checking are specified by an ordinary DRC script of EXPERT/GUARDIAN DRC. The violations are not saved in the DRC error database, but are reported by blinking markers on the screen. If the cursor is placed at a blinking error marker, a short description of the error appears as a small pop-up label.

Note: The checks are performed when object creation or modification is completed. This prevents cluttering by false errors during the (uncompleted) operation.

Note: Real-time DRC takes an advantage of multi-CPU machines. Editing operations are performed using one processor while Real-time DRC loads another one. As a result, you will not feel a slowdown when editing with DRC on guard.

Note: On a single-CPU machine, Real-time DRC runs in background mode with user-definable priority for processor resources.

Set Up and Control

Real-time DRC script and priority of Real-time DRC with respect to editing operations can be set up using **Real-time DRC Setup**. To invoke **Real-time DRC Setup** dialog, open the DRC script panel, load a DRC script, and run **Setup→Real-time DRC Setup** menu command in the DRC Script panel.

Verification→DRC→Real-time DRC submenu provides the following commands to control Real-time DRC:

- **Enable** menu item allows you to enable/disable execution of Real-time DRC with parameters previously set in the Real-time DRC Setup. Once Real-time DRC is set up, you may enable/disable it any number of times. By default, Real-time DRC is disabled. Disabling Real-time DRC clears it of all previous error reports.
- **Show Error Flags** menu item allows you to make errors reported by Real-time DRC visible/invisible.
- **Show Critical Zone** menu item is only available if the **Show Error Flags** menu item is checked. It allows you to make critical zones visible/invisible, and helps to resolve DRC violations according to rules defined by DRC scripts (see Figure 14-5).
- **Check in Area** menu command allows you to perform a special version of **DRC In Area...** command. This command works almost the same way as the **DRC in Area...** command. The only difference is in the scope of the check as described in "Real-time DRC Reports" on page 14-6.

The **Real-time DRC** bar provides quick access to Real-time DRC Setup and Real-time DRC menu commands. Real-time DRC Bar can be made visible by checking the corresponding box in the menu command, **View→Toolbars→Real-time DRC**.

To use Real-time DRC you should first:

- run Real-time DRC Setup
- enable Real-time DRC

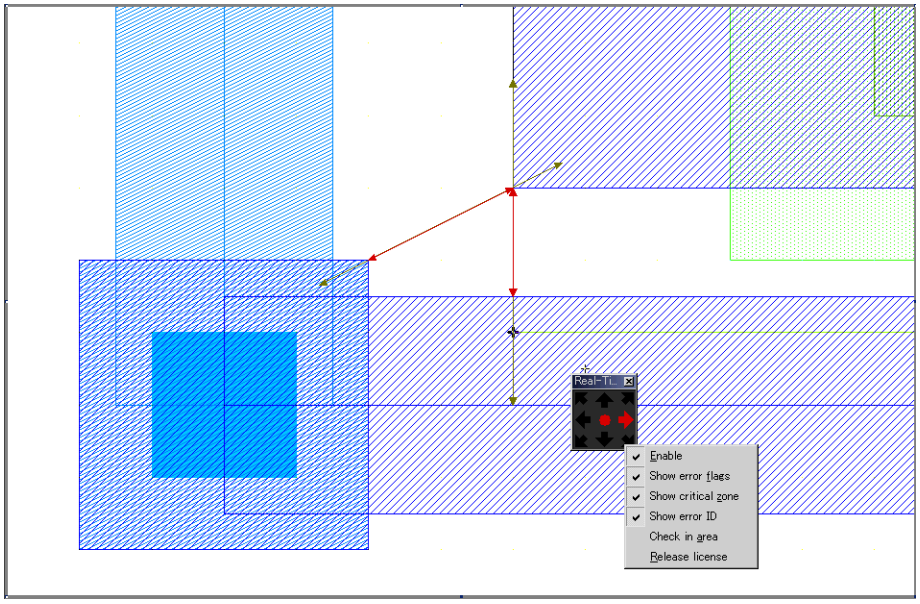


Figure 14-5: Real-time DRC Bar, Error Flags and Critical Zones

Real-time DRC Reports

When creation/modification of an object is finished, Real-time DRC performs checks with this object.

The scope of checks is defined as follows:

- For primitive objects (boxes, wires, regions), checks are performed against the remaining geometry of the current cell and the object itself.
- For a cell instance, checks are performed against the geometry of the current cell, and against the geometry of other instances. Real-time DRC does not report errors between segments that both belong to the same cell instance (anywhere in its hierarchy).

Violations are marked by blinking double-headed arrows. If the cursor is placed at one of these markers, a description of the error is given. **Real-time DRC** bar contains an **Error Indicator**, which helps locate the errors if they are off-screen. The **Error Indicator** has nine fields (3x3 checkerboard). The red blinking dot in the central field means that you have errors on screen. A red blinking error in any of the remaining fields shows the direction when there are off-screen DRC errors.

Hint: If you have a red dot, but see no error markers, this means that your resolution is too low. You must zoom in, possibly, several times, until you see the error markers, or until the error indicator shows you that there are errors off-screen.

There is a switch to show/hide error markers.

Disabling Real-time DRC clears all collected error reports. When layout view becomes inactive, error reports in this view are also cleared.

14.3: Guardian LPE Netlist Extractor

GUARDIAN LPE is a fast and memory efficient tool for extracting Spice netlist from IC mask layout running on PC/Windows NT platform.

If a free license for GUARDIAN LPE is available, its functions are accessed from the **Verification→Netlist Extraction** submenu of the EXPERT layout processor.

GUARDIAN LPE extracts devices (MOSFETs, BJTs, diodes, resistors, etc.) and connectivity information from the layout and produces the circuit description in Spice netlist format. The extracted netlist of the designed chip can further be compared with the netlist produced by the schematic of the design by GUARDIAN LVS layout versus schematic (LVS) comparison tool.

Also, GUARDIAN LPE can call Hipex-RC to generate another SPICE netlist, including parasitic resistances and capacitances. The extracted netlist can be used within a circuit simulator, such as SMARTSPICE, to evaluate the effect by those parasitic RC.

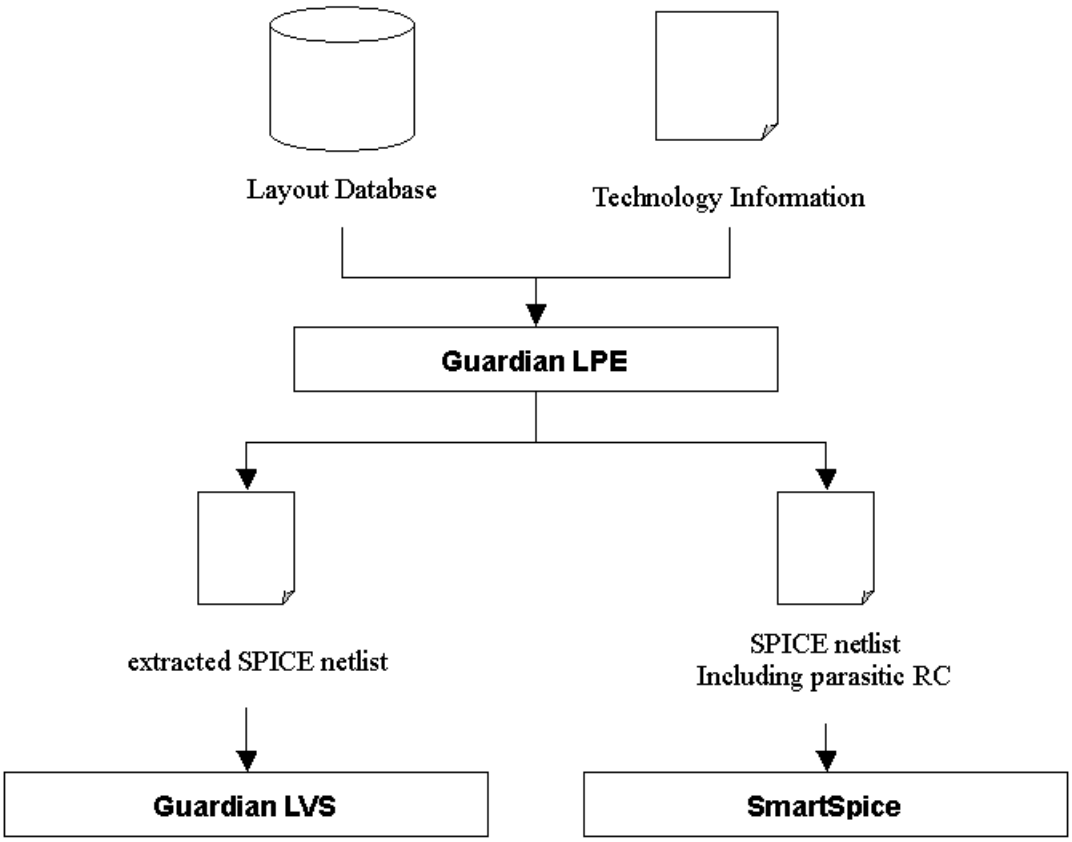


Figure 14-6: Flow-chart of Netlist Extraction and the Post Processes

Device recognition is the process of identifying devices to insert their instances into the netlist representing the extracted version of the circuit.

Each instance of any device is recognized by the existence of a separate shape on a specific layer. This layer will be referred to as a device recognition layer or simply **recognition layer**. It is common to use boolean and/or select operations to create such a layer for each device. For instance, for the most common technologies, MOS field effect transistor is recognized as a shape in the layer that is produced by the overlapping of polysilicon and diffusion layers. Such a layer is often just an artificial idealization, which never exists in a real chip and serves for extraction purposes only.

Apart from device recognition layer, a set of **pin layers** should be assigned to each kind of device. The particular number of pin layers depends on the type of that device (e.g., MOSFET has four pins) while the resistor has two pins. Each of the pin layers has to be listed in at least one of the connectivity sets. See Chapter 3: "User Interface", Section 3.2: "Command Execution" for more details.

In addition to active devices and connectivity information, GUARDIAN LPE also extracts device parameters, such as length and width of MOSFET channel, which have a major impact on device performance.

After the extraction is performed, it is possible to search devices and nets in the layout, as well as to recognize to which device/net a given geometric shape belongs. These functions are accessible from **Verification→Node Search** and **Verification→Node Probing** submenus and the corresponding toolbars of EXPERT.

GUARDIAN LPE reports **electrical rule** violations: **shorted nets** and **broken nets**. They may be inspected on the layout and promptly fixed.

After fixing electrical rule violations and re-extraction of the netlist, one may launch the GUARDIAN LVS tool to compare the extracted netlist against the one produced by a schematic capture tool or extracted from another layout.

The complete description of GUARDIAN LPE can be found in the GUARDIAN LAYOUT VERIFICATION USER'S MANUAL.

14.4: Guardian LVS Netlist Comparator

The GUARDIAN LVS netlist comparison system (LVS) is integrated within CELEBRITY CAD. It may be run both as standalone application, and in batch mode from the operation system's command line. The GUARDIAN tool may be launched from the EXPERT layout editor, as shown in Figure 14-1, if the licence for GUARDIAN is available.

GUARDIAN is a tool used for comparing two circuits as described by their netlists in the standard spice netlist format. One of these netlists usually corresponds to the schematic of a circuit produced (e.g. by the GATEWAY schematic editor). It represents the logic of the circuit. The other netlist is extracted from the circuit layout by the GUARDIAN LPE netlist extractor. It represents the actual physical layout of the circuit as produced (e.g. by EXPERT layout editor). This kind of circuit verification is called LVS (layout versus schematic comparison).

It is not uncommon to compare two schematic netlists (SVS or schematic versus schematic) or two layout netlists (LVL or layout versus layout).

GUARDIAN compares circuits that may contain both primitive devices (flat designs) and hierarchical blocks (cells in the layout or subcircuits in the schematic) of different kinds (hierarchical designs).

GUARDIAN uses the Spice netlist format for comparison. Spice netlists describe circuits at the device level (i.e. in terms of transistors, diodes, resistors, and capacitors), with hierarchy represented in terms of subcircuits and their instances.

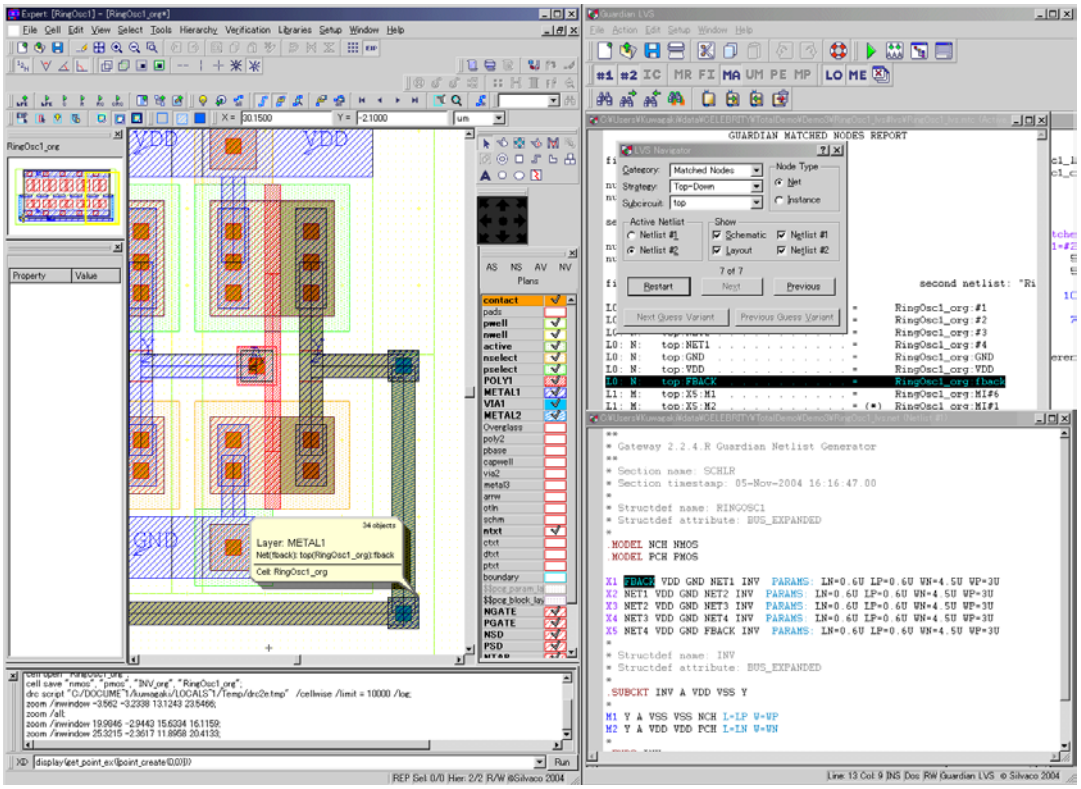


Figure 14-7: Cross Probing between Expert and Guardian LVS

The basic flow of the netlist comparison is as follows:

GUARDIAN performs **Device Reduction**. At this stage, series and parallel devices are merged and unused devices are filtered out, if the corresponding options are ON.

GUARDIAN then performs **logic function recognition** and reduces groups of transistors to "block"

with permutable (interchangeable) terminals.

Finally, the actual **comparison** takes place, at which time GUARDIAN attempts to **match** elements of the two (reduced) netlists against each other.

Both preprocessing and comparison are controlled by various options and conditions. Options are specified through user interface or control files. An example of the latter option is the list of prematched nets, devices, or subcircuits.

The results of the netlist comparison (filtered, merged and reduced devices, matched pairs, and discrepancies) are stored in report files. In the report of unmatched nodes, GUARDIAN gives suggestions about nodes that could have been matched if discrepancies between netlists were fixed.

LVS reports can be cross-probed in both netlist, as well as in the layout, if Simucad's intertool communication (SIPC) is enabled.

The complete description of GUARDIAN can be found in the GUARDIAN LAYOUT VERIFICATION USER'S MANUAL.

14.5: Node Probing

14.5.1: Overview

This operation highlights (selects) all layout objects electrically connected with a given object. It is designed to trace electrically connected nodes throughout the hierarchy of a cell. Also, it allows using the short locator, which generates the shortest pass for specified points in the highlighted node, to find the reason of the short.

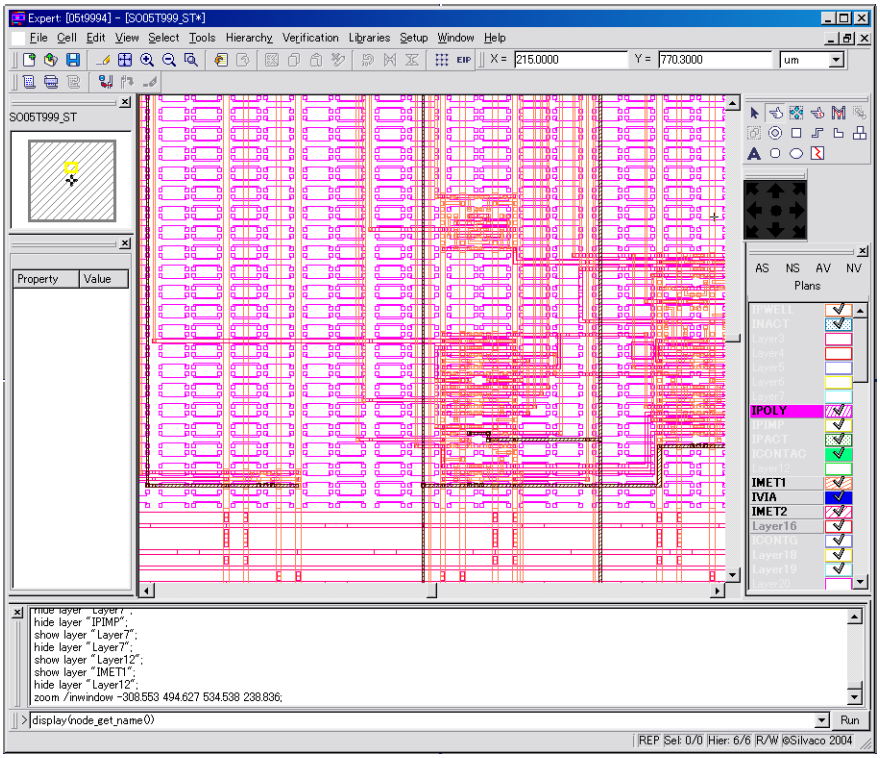


Figure 14-8: Node Probing in Expert

Before probing a **node**, layer-to-layer connectivity has to be assigned as described in Chapter 11: "Setup", Section 11.4.4: "Connectivity Setup".

When the menu command **Verification→Node Probing→Pick Node** is selected, the cursor assumes the node probing shape. You must point and click at the geometric object whose electric connections are to be traced. As a result, the whole net electrically connected to the probed object will be highlighted, and you may inspect it using other commands from **Verification→Node Probing** submenu. The corresponding commands can also be executed from the **Node Probing** bar. (This bar may be turned on/off by the **View→Control Bars** command).

The probed object may be anywhere in the hierarchy.

To facilitate the choosing of the required object, preselection highlighting is used. You can also hide some layers for better selectivity. Node probing function ignores layer visibility.

Multiple Nets can be highlighted simultaneously

If the **Shift** key is depressed while the **Probe Node** or **Find Net by Name** tools are activated, the picked net becomes highlighted and available for inspection, and the previously selected nets stay highlighted on the screen.

Node objects from non-selectable layers are always displayed in a wireframe mode

When **Probe Node** or **Find Net by Name** tools are activated for power or ground nets, big highlighted objects from substrate layers often make other node objects hard to see. To see power and ground net objects clearly, set the substrate layers the non-selectable by pressing **Ctrl** while clicking on the layer name, or using the layer plan, where substrate layers are not included or selectable. Now, all non-selectable layer objects are displayed in wireframe mode independently from the filling mode of the probed node set by **Verification→Node Probing→Node Filling** submenu: wireframe, solid or stipple.

Note: The layer of the picked object must be present in connectivity definition.

Note: When a node is being probed for the first time for an unchanged layout, a preprocessing takes place to extract node connectivity and build connectivity index for fast search and display. After preprocessing, the probing of all other nodes for the same layout works faster.

14.5.2: Short Locator

The **Verification→Node Probing→Short Locator** allows you to create the shortest pass connecting more than two specified points in the highlighted node. If you click more than two points that are not of the same potentials, the created shortest pass would include the reason of for the short. If you cannot find the reason for the short, you can reset the points and reduce the target area.

The shortest pass is created in the layer "ShortPass" as a normal wire. If you want to delete it, select it by **Select→By Click / Drag**, and then run **Edit→Delete**.

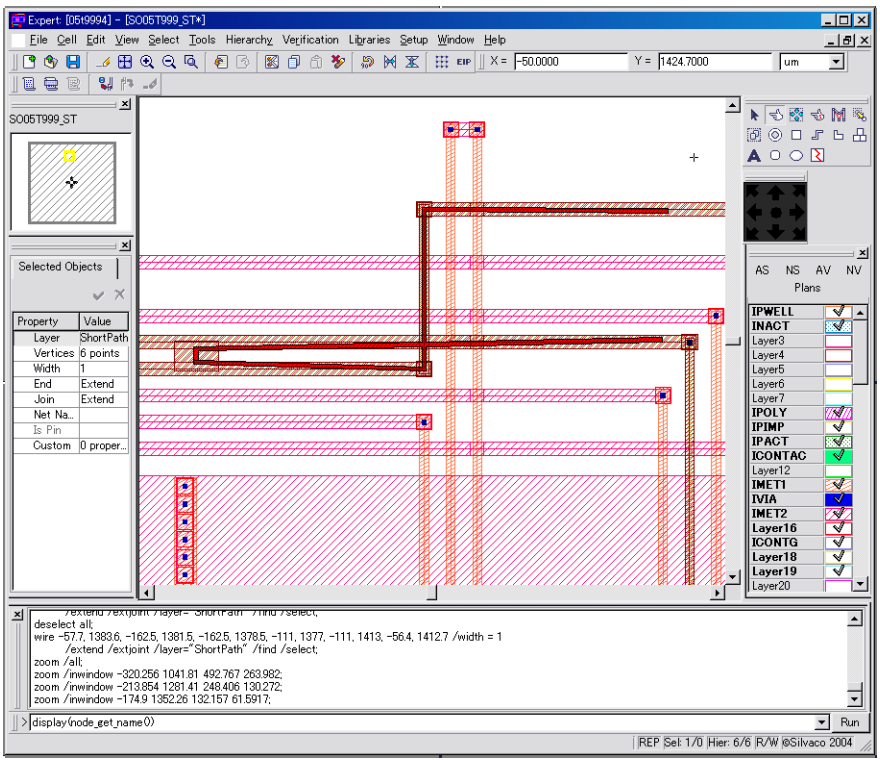


Figure 14-9: Short Locator after Node Probing

The created shortest pass depends on option **Verification→Node Probing→Short Locator Options**.

The **Verification→Node Probing→Short Locator→Find Short** command automatically run the **Short Loacator** tool between text labels in selected nodes, and shows where the short most likely accured.

14.5.3: Node Probing Submenu

The **Verification→Node Probing** submenu contains commands to navigate through the node and to display it.

Pick Node: Described above. If you hold **Shift** during the **Pick Node** operation, the previously shown node remains, and a new node added.

Show Node: Shows/hides the marker that covers the last probed node.

Zoom to Node: Shows the node to fit the window.

Zoom to Object: Zooms to the current object of the node, highlighted by node navigation commands.

Object Descriptors: If this option is checked, the current node object is supplied with a bubble containing its description.

First, Next, Previous, Last Object: These commands highlight the consecutive objects of the node.

EIP Object: This command edits in place the cell that contains the highlighted object of the node.

View in Cell: This command opens a new window with the cell that contains the highlighted object of the node.

Reextract Connectivity: This command rebuilds node connectivity data. Usually it is applied after editing the layout or after changing the technology.

Reextraction Options: This submenu tells you what to do if the project was changed. You have the following options:

- Always Reextract
- Never Reextract
- Prompt for Reextraction

Node Filling:

- Wireframe
- Solid
- Stipple
- Select Stipple…: This command allows you to select the filling pattern for node highlighting during node probing/search. Recall that the color for node highlighting can be selected at the **Setup→Hotkeys/Menus/Colors→Color form**.

Note: If layer type **Selectable** set to off, objects from this layer are always drawn unfilled. This is continent for layers as substrate or bulk.

14.6: Calibre

EXPERT can be used in conjunction with Calibre DRC and LVS verification tools. You can launch verification from within EXPERT, and then use EXPERT for highlighting errors.

Setup

For setting up Calibre tools, please refer to Mentor Graphics documentation. EXPERT expects the `MGC_HOME` environment variable to be set, and pointing to the location where Calibre is installed.

You can also use the environment variable `MGC_CALIBRE_LAYOUT_SERVER` to specify socket number, which will be used for communication between EXPERT and Calibre tools. By default, EXPERT uses socket 9189. If it is not available, EXPERT tries to open different ports. When EXPERT successfully opens a socket, it reports the socket number to the log file.

Calibre DRC

You have two options invoking Calibre DRC: directly from EXPERT as a batch job, or through Calibre Interactive.

To invoke Calibre DRC as batch job, select the menu command **Verification→Calibre→Batch DRC**. You will be prompted with the dialog shown in Figure 14-10.

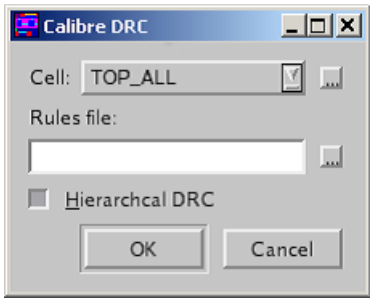


Figure 14-10: Batch DRC parameters

This dialog allows you to select the which cell, and the rules file and mode that DRC will run. To start verification, enter all parameters and click the **OK** button. When the DRC operation finishes, EXPERT will automatically load DRC errors.

To run DRC through Calibre Interactive, select the menu command **Verification→Calibre→Run DRC**. You will be prompted with the Calibre Interactive panel, where you can enter all DRC parameters and start a DRC operation. After DRC finishes, Calibre Interactive will invoke Calibre RVE. Use it to browse DRC errors and highlight them in EXPERT.

Calibre LVS

To run Calibre LVS verification, select the menu command **Verification→Calibre→Run LVS**. EXPERT will launch Calibre Interactive, where you can enter all LVS parameters and start LVS verification. After LVS finishes, Calibre Interactive will invoke Calibre RVE. Use it to browse LVS errors and highlight corresponding layout geometry.

For more information on using Calibre Interactive and RVE, please refer to Mentor Graphics documentation.

Chapter 15: Plotting

15.1: Plotting Layouts

The group of plotting commands from the **File** submenu allows you to plot the visible part of the layout on local or network printers (plotters). You must have the device driver for the required plotter installed. If you do not have the driver for your plotter (especially if you are using some older plotters like Calcomp, Tektronix), you may search internet for manufacturer's web site. Usually you may download drivers from there. Moreover, it is a good idea to download the latest version of the plotter/printer driver even if you already have one installed.

In addition, most Windows NT printer drivers provide redirection of the output data into a file. This file contains control commands recognized by the associated plotter/printer. It gives the option (after the installation of several different plotter/printer drivers on EXPERT'S computer) to generate command files of various formats (e.g., Postscript). These files may then be transferred into other environments and processed by device-specific utilities available in those environments.

This chapter describes the following commands:

- **File→Printer Setup Dialog**
- **File→Page Setup**
- **File→Print Preview**
- **File→Print**
- **File→Print Status**

15.2: Printer Setup Dialog

This standard **Printer Setup** dialog appears on first using of commands **File→Print Preview** and **File→Print**. It can also be called from the **Page Setup** dialog. **Printer Setup** dialog is used to assign the output device (printer name), size and orientation of paper sheets, as well as other device-specific properties. These properties are used for previews and plotting operations. You will only see the required plotter/printer name in this setup if it is properly installed in your system; otherwise, contact your system administrator for assistance.

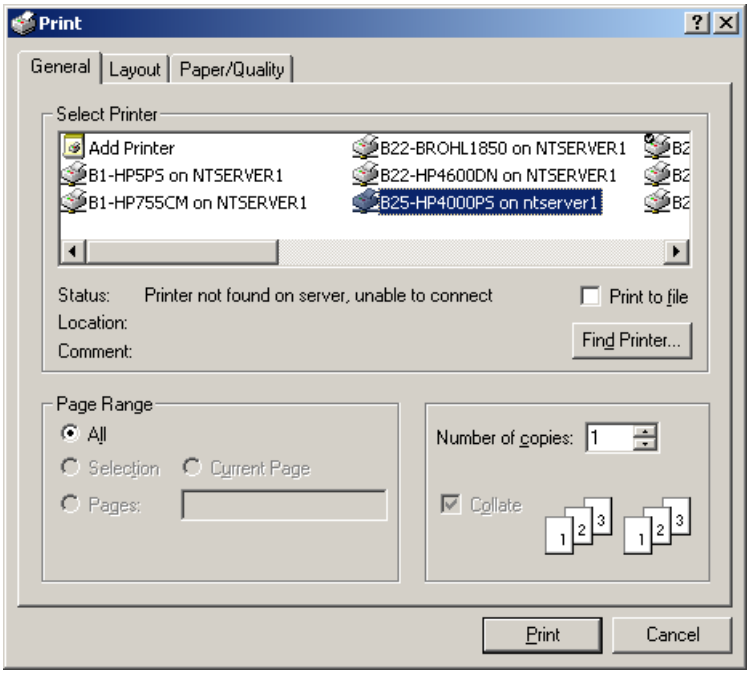


Figure 15.1: Printer Setup

15.3: Print Preview

The menu command, **File→Print Preview** invokes EXPERT'S **Page Setup** dialog followed by the **Print Preview** window to show the arrangement of the plotted layout on paper sheets.

Opening preview windows can take long time, and may be cancelled by hitting the **Esc** key.

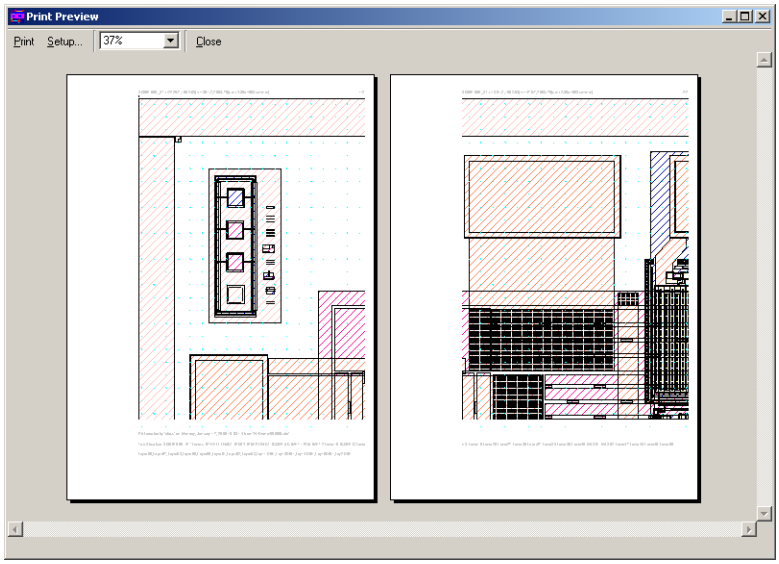


Figure 15.2: Print Preview

15.4: Page Setup Dialog

This dialog is invoked by the execution of the menu commands **File→Page Setup**, **File→Print Preview** and **File→Print**. It consists of three sheets to specify different classes of parameters: **Page Layout** sheet, **Plotting Style** sheet, and **Miscellaneous** sheet, which are described below.

15.4.1: Page Layout Sheet

This sheet is used to define the size of the plotted image, in particular, to split the drawing into several pages, and to set drawing margins on the page.

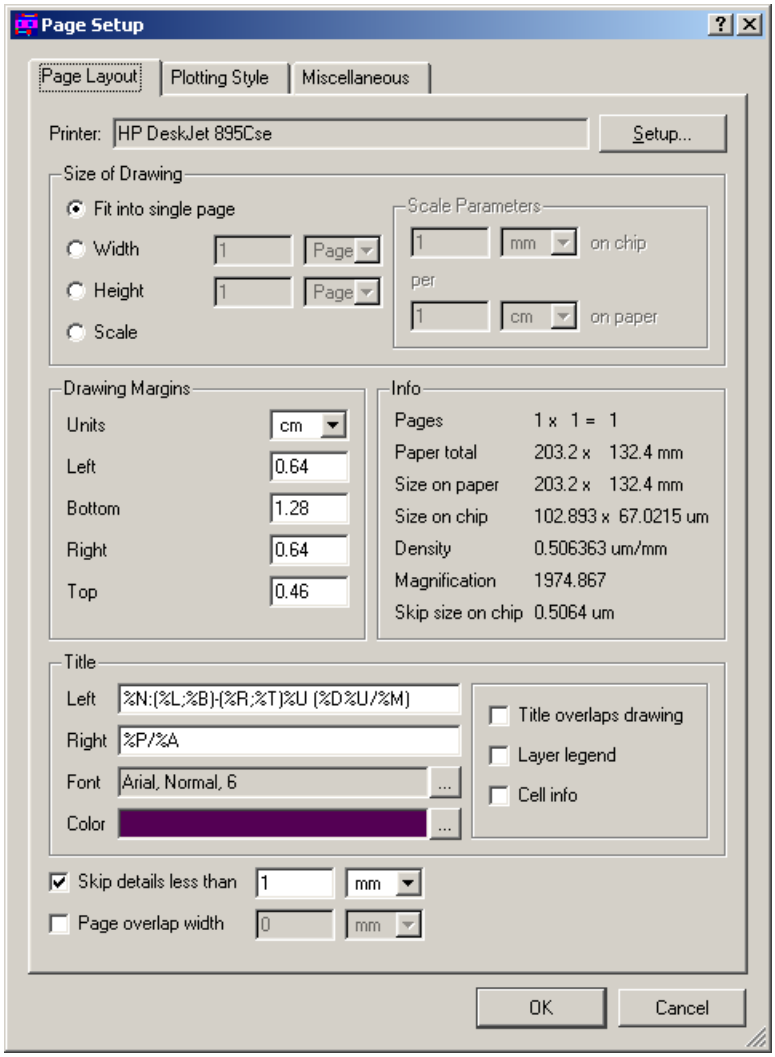


Figure 15.3: Page Setup: Page Layout

Sizing

The simplest option for sizing the image is **Fit into Single Page**. If it is selected, the layout visible on the screen fits onto a single sheet of paper.

Hint: You may wish to set the **Landscape** option from the Printer Setup, together with the **Fit into Single Page** option, because usually the height of the screen is less than its width.

The **Width** field defines the horizontal dimension of the plotted image. The height of the image will be computed automatically to provide the same scaling along both axes.

The **Height** field defines the vertical dimension of the plotted image. The width of the image will be computed automatically to provide the same scaling along both axes.

The **Scaling** field defines the magnification of the plotted image. A sample reading in this dialog is: “Scaling 50 nm per 1 mm on paper”.

The **On chip** field selects the unit of measurement and the size in the actual chip layout. The **On drawing** field selects the unit of measurement and the size in the drawing. In this way you may specify both direct and inverse scaling.

The location of the plotted image on the drawing media page can be defined by layout margins. The **Left**, **Bottom**, **Right** and **Top** fields defines the margins near the corresponding page borders. Margin size is expressed in units selectable in the **Units** field.

Images drawn on consecutive pages may be overlapped by the size defined in **Overlap width** field. With this goal check in the appropriate box, the **Title overlaps drawing** box controls page space for the title. If it is not checked, the title and layout are drawn in separate spaces.

Data units can be expressed in: `nm`, `um(mcm)`, `mil`, `mm`, `cm`, `inch`, `m`, and user units `(Lambdas)`.

Media units can be expressed in: `mm`, `cm`, `inch`, `m`, and pages.

The sizing information resulting from the selected settings is shown in the **Info** group of this sheet.

Customizable Title

The **Left** and **Right** fields of the **Title** group defines the custom title line on plot pages.

Part of the title is right aligned.

These fields are patterns that can contain any user text, together with control symbols. The latter ones are substituted by actual values taken from the layout data. The following control symbols are possible:

```
%N  - Current cell name
%L  - left coordinate of data plotted on the current page
%B  - bottom -"-
%R  - right  -"-
%T  - top    -"-
%D  - plot density
%W  - page width
%H  - page height
%P  - page number
%A  - total count of pages
%C  - name of units of measurement for size on chip
%M  - name of units of measurement for size on paper
%U  - name of units of measurement in the current Expert’s window
```

Example

```
(Left part): Cell %N; density: %D %C/%M. Plotted by Nikk.
(Right part) Pg. %P of %A
```

The **Font** button selects the font type, size, style, color, etc., for the plotted title from fonts supplied by your system.

The **Title overlaps drawing** option is self-explanatory.

Other

Skip details less than <Size>: Where **<Size>** is the minimum feature size on the plot.

The objects with height and width less than **<Size>** will not be plotted.

Page overlap width: The amount defined in the **Overlap width** field overlaps the images printed on adjacent pages. If the overlapped width is a non-zero, then page registration marks in the shape of crosses will be plotted. These marks are used for exact continuation of the plot across the adjacent pages.

Title overlaps drawing: This option saves some space if the top part of the drawing is unimportant.

Layer Legend and **Cell Info** options are to put the corresponding information on the plot.

15.4.2: Plotting Style Sheet

You may reassign plotting style as a whole by loading a style file, or layer by layer. The layers for which you may reassign plotting style are shown in the list in the center of the **Plotting Style** sheet of the **Page Setup** dialog. If you do not see any layers in the layer list, click the **Merge** button. This puts the layers of the current technology into the layer list for plotting style setup.

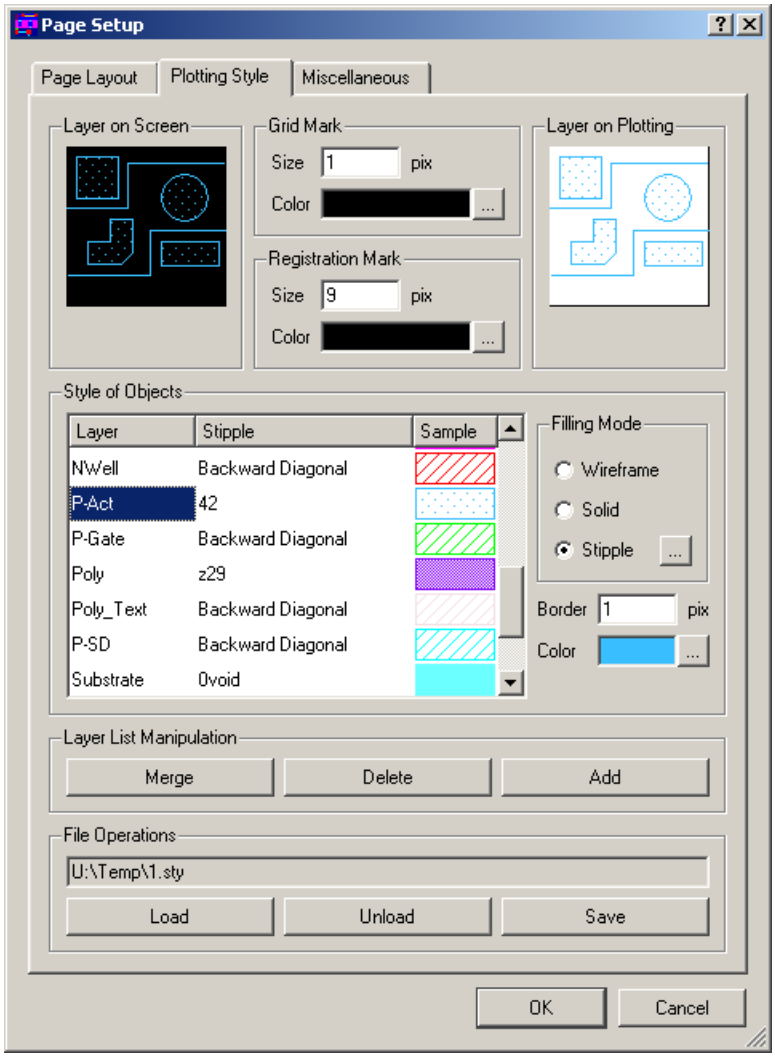


Figure 15.4: Page Setup: Plotting Style Sheet

File Operations

You may load and save plotting styles to/from files with the `.sty` file extension. To start a new style file, select the **Unload** command. It clears the current style setting.

Layer Operations

Merge: Adds all layers from the current project to the layer list shown on the Plotting Style sheet. Only layers absent in the latter list are added.

Delete: Deletes the highlighted layer from the style list.

Add: Adds a new layer to the style list. If this layer is present in the project, its screen style settings are copied to plotter style settings.

"Style" Group

Click at the layer name if you want to change its appearance on the plot. Two previews in the top part of the **Plotting Style** sheet appears: **Layer on Screen** and **Layer on Plotting**.

You may select one of three drawing styles for the highlighted layer by clicking at the corresponding radio buttons on the right side of the **Plotting Style** sheet:

- wireframe
- solid
- stipple

In the latter case, you are prompted to select a particular stipple style. There are three stipple style categories:

- bitmaps
- system stipples
- X-stipples

System stipples (called hatches in previous versions of EXPERT) are usually much faster than bitmaps. X-stipples are also very fast.

Stipple patterns are defined by bitmaps. There are a number of predefined stipple bitmaps in EXPERT. Moreover, EXPERT has a stipple editor to create fully customized stipple bitmaps. The stipple editor is called from the **Layer Setup** dialog (**Setup→Technology→Layer Setup** command).

Color: Selects the color for plotting the layer. Often colors on the plotter differ from those on screen. To handle this situation, see Section "Layer Color Setup".

Cell Instance Plotting Style

Cell instance plotting style is selected exactly in the same way as for layers when `***Instances***` item is selected in the list.

Other

Border Thickness: It is possible to specify the thickness of the borderline of objects in the given layer. Thickness is specified in pixels.

Registration Marks: Cross-shaped registration marks are plotted. It is possible to specify their color and size on the Plotting Style sheet.

Plotting

Grid and Supergrid: Both grid and supergrid are plotted if they are present on the screen. It is possible to specify their color and size on the **Plotting Style** sheet.

15.4.3: Miscellaneous Sheet

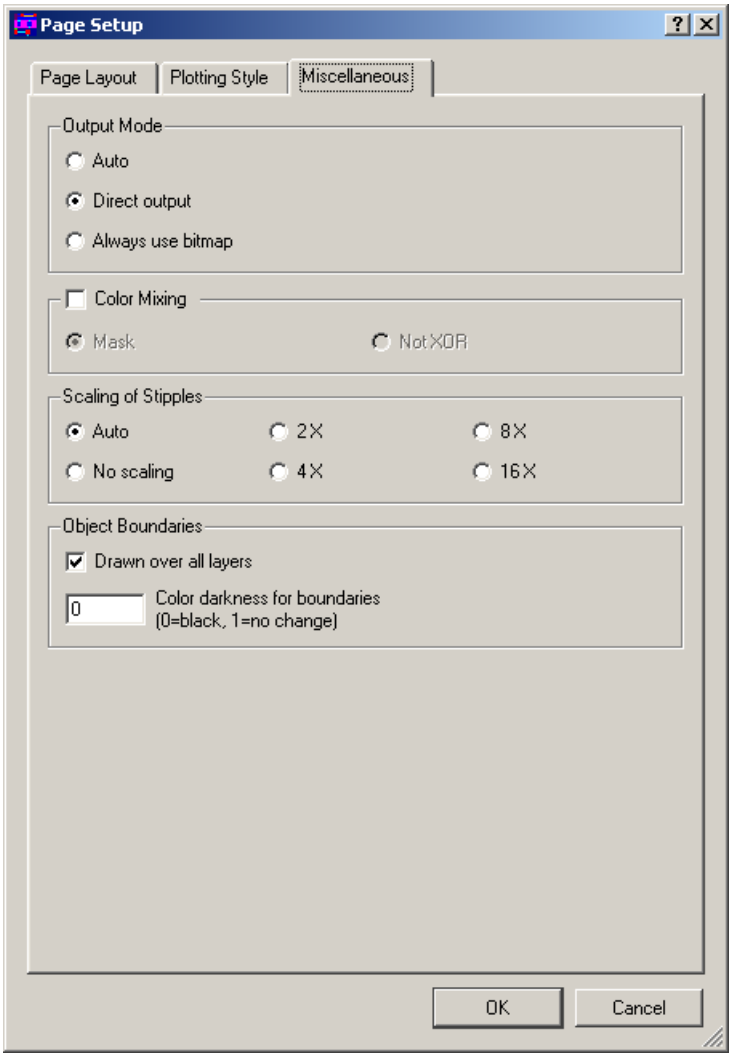


Figure 15.5: Page Setup: Miscellaneous Sheet

This page contains the option **Output Mode**. There are three available choices:

- **Auto:** EXPERT will try to select most appropriate output method.
- **Direct output:** Layout will be plotted using the printer device driver.
- **Always use bitmap:** All plotting operations will be performed in memory by EXPERT, and then the prepared bitmap will be transferred to the printer device.

In **Direct Output** mode:

- Several operations with plot (such as color mixing, transparency, etc.) are performed directly by the plotter's driver. Otherwise, some preliminary operations are performed by EXPERT.
- EXPERT requires less memory;
- The amount of data transferred to the printer depends on the amount of graphics on layout, so plotting may be slower for large plots.

In **Bitmap Output Mode**:

- Such operations (color mixing, transparency, etc.) are performed by EXPERT and may be more accurate.
- EXPERT requires more memory;
- The amount of data transferred to the printer depends on page size, number of pages, dpi and color depth. This amount does not depend on the amount of graphics on layout, so plotting may be much faster for large plots.

Try different output modes to see which one suits your printer/plotter in terms of plotting quality and speed.

The other settings of this panel make it possible to do the following:

- Draw layers in color mixing mode.
- Draw object boundaries on top of all drawings, so that they will come sharp in stipple mode.
- Change boundary darkness from black color up to current color by setting a real number from 0.0 up to 1.0, when **View→Filling→Stipple** or **View→Filling→Solid** modes are set up.
- select magnification of stipple patterns.

15.5: Print

The **Print** command invokes the following dialogs in turn:

- The standard **Printer Setup** to assign the output device used for plotting and to change printer device parameters;
- EXPERT'S **Page Setup** dialog, described in the corresponding section,

Then it prints the visible part of the layout. The printing session may be interrupted by hitting the **Esc** key.

In addition, most Windows NT printer drivers provide redirection of the output data into a file. This file contains control commands recognized by the associated plotter/printer. It gives the option (after the installation of several different plotter/printer drivers on the EXPERT'S computer) to generate command files of various formats (e.g., **Postscript)**. These files may then be transferred into other environments, and processed by device-specific utilities available in those environments.

15.6: Print Status

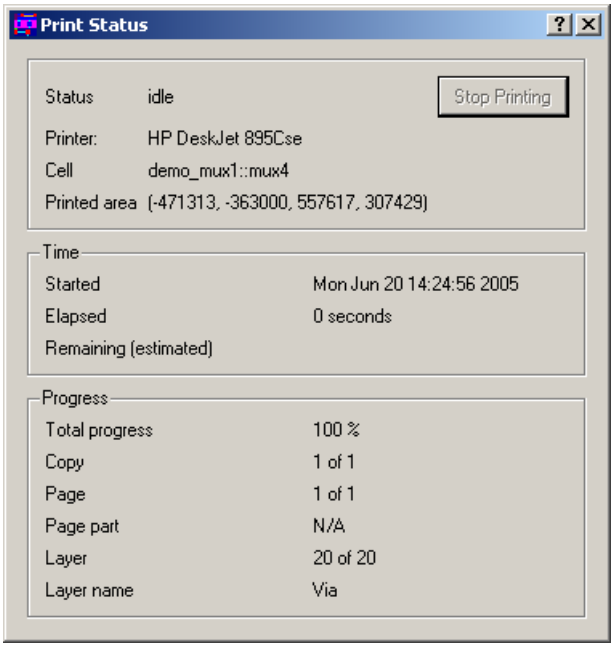


Figure 15.6: Print Status

The **Print Status** command shows the **Print Status** window which displays the parameters and status of current printed cell. You can cancel printing by clicking the **Stop Printing** button.

When printing is finished the **Print Status** window displays the status of the last printed cell.

[This page intentionally left blank]

Chapter 16:
Libraries and Version Control for a Multi-User Design Environment

† 16.1: Overview

This chapter describes the features of EXPERT that provide access to a common design database in a multiuser environment.

The following topics are covered:

- Library setup for a work project;
- Shared and non-shared libraries;
- Shared cells management;
- User Access rights.

† 16.2: Libraries

Library: A file(s) in EXPERT'S layout database (`*.eld`, `*.sld`) format that contains a number of layout cells; possibly a complete hierarchical project. There are two types of libraries supported by EXPERT:

- **Non-shared library:** This library consists of one `.eld` file which contains all library cells.
- **Shared library:** The library of this type consists of one `<library_name>.sld` file and a lot of (one per cell) `<cell_name>.eld` files in a `<library_name>.cll` subdirectory.

Non-shared library: Libraries of this type can be opened in **exclusive** or **read-only** mode. Gaining exclusive access to such a library provides full control over its contents, and you can view and modify cells from this library and use them in the project. No one else can add, delete or modify any cell of this library.

There are three possible cases how a non-shared library can be opened:

- If you do not have the write permissions to the `<library_name>.eld` file, it will always be opened in **read-only** mode.
- If you have the write permission, but the library is currently opened in exclusive mode by another user, it will be also opened in **read-only** mode.
- In other case, if you have the write permission, and nobody else has opened this library in exclusive mode, it will be automatically opened in **exclusive** mode.

To change file access permissions, please contact your system administrator.

On MS Windows operation systems you can change the read-only property of `<library_name>.eld` file to disable it's editing or opening in exclusive mode. To do that, open the File Properties dialog which is available from file context menu in explorer.

Shared library: Libraries of this type may be accessed by many designers. If you have proper access rights for a shared library, you can check out cells from it for editing. Only one user can edit any particular cell at a time. The changes become visible to the other users after the cell is saved or checked in.

The **Libraries→Update Library** menu command allows you to update the current library after other users have saved or checked into this library.

A shared library can be created on local and network drives.

Both **shared** and **non-shared** libraries can be opened in **read-only** mode. Cells from libraries opened in read-only mode may be viewed and used in other projects by many users simultaneously; however, no one can modify them.

† 16.3: Typical Scenarios for Library Usage

There are several levels of sophistication of library usage:

- You can use non-shared libraries with full access to all cells in all these libraries.
- You can use libraries with or without version control.
- Version control may be performed on a single computer, or across the network by simultaneous access of multiple users.

16.3.1: Non-Shared Libraries

Non-shared library is a single `<library_name>.eld` file which contains library technology and all library cells inside.

Non-shared libraries can be opened in two modes:

- exclusive;
- read-only.

The **basic flow** for the usage of such libraries is as follows.

1. Create an ordinary project file in `*.eld` format. Suppose its file name is **mylib.eld**.

2. Open your work project or start a new one. Be sure that it uses the same technology as **mylib.eld**.

3. Set the **mylib.eld** library as follows:

A. Run **Libraries→Setup...** menu command.

B. Click **Add** button at the **Library Setup** panel.

C. Click the **file browser** button (the one with three dots (**...**)), select **mylib.eld** on your disk and click **Open**. You will see that the "logical name" field of the **Add Library** panel will be filled by default value. You can change the logical name or leave it as is and click **OK**.

D. To use library cells in your project, you must **activate** it by the corresponding button.

Now, if you open a cell browser of any EXPERT'S editing command, you will see **mylib** in the library list, and you can open its cells.

16.3.2: Shared Libraries/Cell Version Control

Version control is possible for libraries of the **shared** type.

The **basic flow** for the usage of shared libraries is as follows.

1. Create an ordinary project file in `*.sld` format. Suppose its file name is **mylib.sld**.

2. Open your work project or start a new one. Be sure that it uses the same technology as **mylib.sld**.

3. Set the **mylib.sld** library as follows:
   - Run **Libraries→Setup...** menu command.
   - Click **Add** button at the **Library Setup** panel.
   - Click the **file browser** button (the one with three dots (...)), select **mylib.sld** on your disk and click **Open**.
   - To use library cells in your project, you must **activate** it by the corresponding button.

Now, if you open a cell browser of any EXPERT'S editing command, you will see **mylib** in the library list, and you can open its cells.

16.3.3: Shared Access to Projects in Read-Only Mode

If a designer opens an exclusive project for editing, other designers can open this project in read-only mode. If the first designer performs a "save" operation, the other designers are prompted to reload the project, if they want to see the actual data.

† 16.4: Project/Library Management

16.4.1: Library Setup for Your Project

A Layout cell library is a file in the same format of EXPERT'S layout database (`*.eld`) format; therefore, any library may be opened as a project in EXPERT.

If cells from libraries are used in a work project, these "external" cells are identified by **cell name** and **logical library name**. In xi-script, and in many other parts of EXPERT, you will see such extended cell names in the following format:

```
<logical_library_name>::<cell_name>.
```

For example, `cont::PM1Con`, denotes the cell **PM1Con** from the library with the logical name: **cont**.

A logical name is assigned to a library file through the **Library Setup** panel (menu command **Libraries→Setup...**). See Figure 16-16-1.

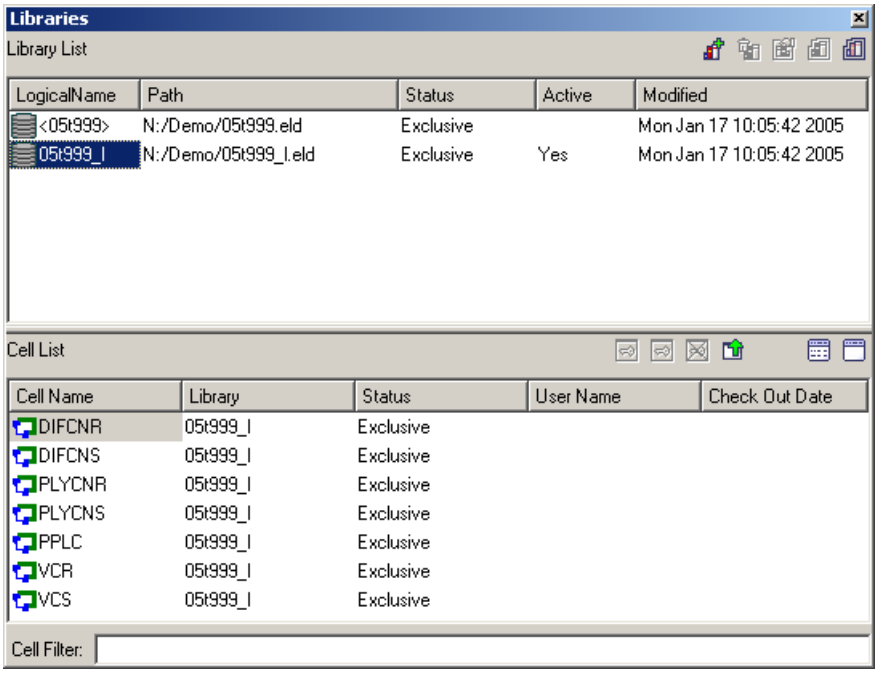


Figure 16-1: Library Setup Dialog

The **Library Setup** panel contains the list of libraries defined for your workspace. If you open a project that contains cells from some library, this library is automatically activated if it was presented in list. All other libraries must be added to this list manually using the **Add** command button.

Logical Name, Path

It is impossible to have the same logical name for two different files in the library setup. No file can have two logical library names. If, for some reason, you want such duplicates, you must use two copies of the same file. Logical names are case sensitive.

Status

This column shows the current library status. It can be one of:

- "Invalid Path" for libraries whose path was not specified or is incorrect;
- "Exclusive" for non-shared libraries opened in exclusive mode;
- "Read-Only" for shared or non-shared libraries opened in read-only mode;
- "Shared" for shared libraries opened in shared mode.

Read Only Status

The "Read Only" status is displayed if the library is opened in read-only mode. For non-shared libraries it means that you can not change library technology and cells inside. For shared libraries it means that you can not change its technology. To check if you can change a cell in the shared library, check the status of the cell.

Active

Any library must be activated before using. Some libraries are activated automatically, for instance, when you open a project which contains cells from those libraries. To activate a library manually press the **Activate** button in the **Libraries Setup** window. To deactivate it, click the **Deactivate** button. You can not deactivate library if it is in use (its cells are instanced in any opened project).

Modified

For non-shared libraries, this column shows the date and time of the last modification of the library or any its cells. For shared libraries, it shows the last modification date and time for the library technology only.

Add Button (Library Setup)

It is used to add libraries into the setup. When you click this button, the **Add Library** dialog shown in Figure 16-16-2 appears.

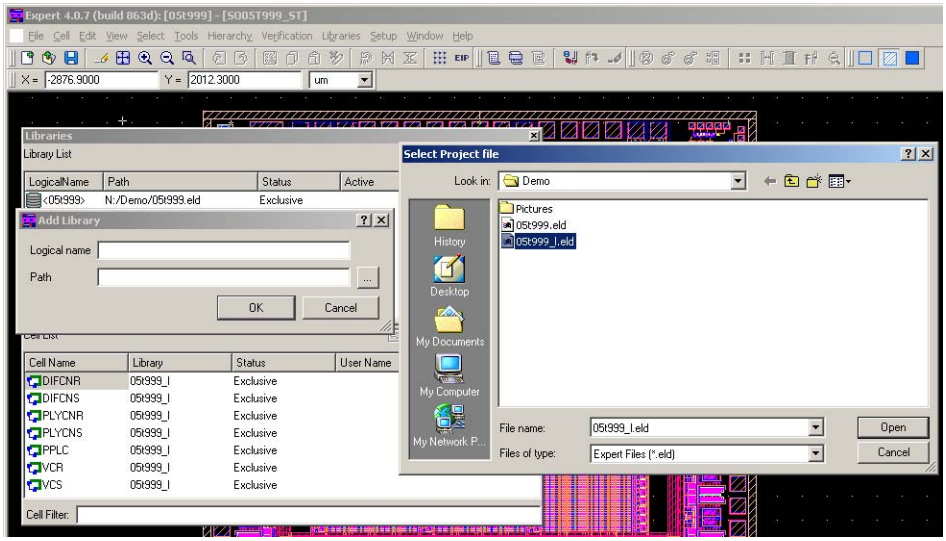


Figure 16-2: Adding a library to setup

You can type the logical name and pathname; however, the file must exist. Therefore, the easiest way is to click at the file browser button (the one with three dots on the right side in the Add library dialog box). You will see an ordinary file browser dialog. If you select a `*.eld` (or `*.sld`) file and click the **Open** button, the selected file selected path will be shown in the **Add Library** dialog box, as shown in Figure 16-16-3.

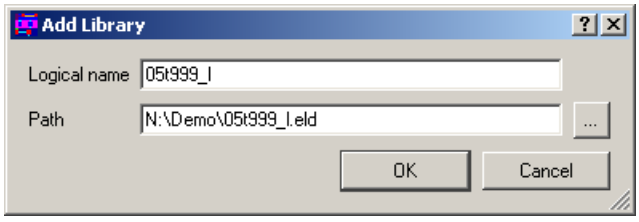


Figure 16-3: Entering library parameters

The default logical name is the file name without the path and the extension. Now you may edit the logical name and type manually.

Delete Button (Library Setup)

If you select a library by clicking at its logical name, and then click the **Delete** button, the library will be deleted from the library setup; however, you cannot delete a library from the setup if some of the cells from this library are used in the opened project. You have to deactivate the library before deleting.

Properties Button (Library Setup)

If you select a library by clicking at its logical name, and then click the **Properties** button, you may change library parameters; however, you cannot do this if some of the cells from this library are used in the opened project.

Activate Button (Library Setup)

If you select a library by clicking at its logical name, and then click the **Activate** button, the library will be activated for the opened project. This means that you may see this library in cell lists of various operations of EXPERT, and hence you may use cells from this library in all editing operations.

If a project has instances of cells from another library, two cases are possible:

1. If this “secondary” library is present in the library setup, and this library is defined correctly, it is activated automatically when the project is loaded.
2. The first library, otherwise, will have unresolved references. If you add the “secondary” library to the setup, these references may be resolved.

Deactivate Button (Library Setup)

If you select a library by clicking at its logical name, and then click the **Deactivate** button, the library will be deactivated for the opened project. This means that access for cells of this library will be lost.

You cannot deactivate a library if some of its cells are used in the opened project.

16.4.2: Library and Cell Access Rights

Access rights for non-shared libraries, shared libraries and cells in shared libraries can be granted by means of operation system only.

It is not possible to assign different access rights for cells inside one non-shared library (`.eld`), but the system administrator can grant required access rights for the whole library.

To assign different access rights for cells inside one library, use shared libraries where every cell is stored in the separate `<cell_name>.eld` file. In this case, your system administrator would be able to assign proper access rights for every cell in the library.

The basic access rights for library or cell are:

- No access
- Read Only

- Change

For Non-Shared Library

- "No access" right means that you can not open this library at all;
- "Read Only" access right means that you can open this library, any cell in this library but cannot add, delete or modify any cells inside;
- "Change" right means that you have full access to the library and its cells, and can add new cells and delete/modify any existing cells.

For Shared Library

- "No access" right means the same as for non-shared library (it can not be opened at all);
- "Read Only" access right means that you can open this library, but cannot change the technology data of this library;
- "Change" right means that you can open the library and modify it's technology.

Note: "Read Only" access right of shared library does not affect to cells inside it. You can have "Read Only" access right for library and "Change" access right for all or some cells inside. These cells can be edited without any restrictions.

For Shared Cells

- "No access" right means that you cannot open or delete the cell;
- "Read Only" right means that you can open the cell in read-only mode, but cannot edit or delete it;
- "Change" right means that you can open, edit and delete this cell.

To create new cells in library you must have file creation permissions in `<library_name>.cll` directory.

Please, contact your system administrator to assign correct access rights for libraries.

Some operation systems have more access right options; they are also supported by EXPERT.

Non-shared library `<library_name>.eld` file, shared library `<library_name>.sld` file and shared cell `<cell_name>.eld` files can be encrypted by means of an operation system to provide more security.

Note: Encrypting and compressing of EXPERT `*.eld` files on fly can significantly reduce the speed of many EXPERT functions.

---

† 16.5: Cell Management

16.5.1: Usage of Library Cells in Projects

To use cells from a library you must do the following.

1. If you want to use a shared library you don't have in your workspace, you must add it:
   - Run the **Libraries→Setup** menu command.
   - Click the **Add** button at the **Library Setup** panel.

Click the file browser button (the one with three dots (...)), select the required library on your disk and click **Open**. The fields of the **Add Library** panel is filled by default values. You can change the logical name and library type or leave them as they are and click **OK**.

2. Activate recently added libraries by clicking the **Activate** button.

Now, if you open a cell browser of any EXPERT'S editing commands, you will see the new logical name in the library list, and you can open cells from this library for various operations, depending on library type and status of a cell.

16.5.2: Cell Version Management

The menu command, **Libraries→Setup**, invokes a panel shown in Figure 16-16-4.

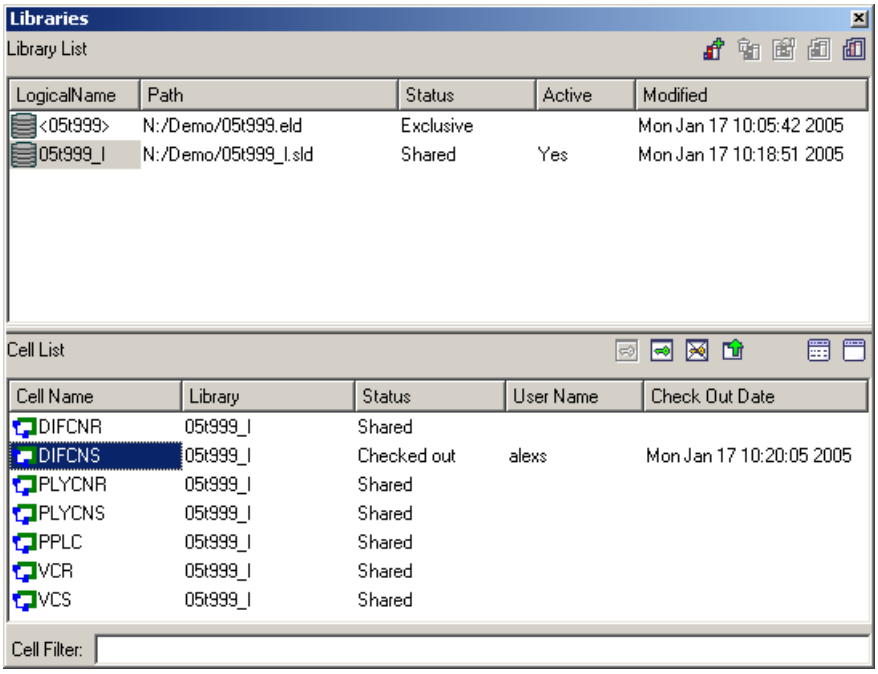


Figure 16-4: Cell Management for Controlled Libraries

Library List and Commands

The **Library List** shows the controlled libraries in the current setup. If you select a library in this list, the list of its cells will be shown in the **Cell List** below. You may select several libraries, and all their cells will be listed below.

The library list may be sorted by any column in ascending or descending order by clicking at the column title button once or twice.

Cell List and Commands

This list shows cells from all active libraries selected in the Library list.

The **Cell** list may be sorted by any column in ascending or descending order by clicking at the column title button once or twice.

The **status** column may contain:

- **Exclusive:** This cell is located in library which was open in exclusive mode. It can be edited without applying the "check out" operation.
- **Shared:** Shared cell, "check out" action required before editing.
- **Read-only:** Cell can not be edited.
- **Checked out:** Cell is shared, checked out. If it is checked out by current user, cell is ready for editing.
- **Unresolved:** Cell is instanced to some project or library cells, but the cell or it's library was not found.

The **user name** column shows the name of the user who checked out the cell.

The **check out date** column shows date/time of the check out operation.

Check out the selected cells from shared libraries. You must check out a cell from a shared library if you want to edit it.

Only one user may check out a particular cell at any given time. The remaining users will see the previous version of the cell until it is checked in.

Cancel check out for the selected cells. These cells remain unchanged in their libraries, and other users may check them out.

Check in the selected cells. Cells are saved automatically before checking in. These cells are changed in their libraries and other users may check them out.

Open: Opens all selected cells.

16.5.3: Edited Cell Version Management

For the edited cell (either opened directly or through Edit in Place (EIP) commands) the following version control commands are available directly from the menu and from the **Version Control Bar**.

Libraries→ Edited Cell→Check Out

Libraries→ Edited Cell→Check In

Libraries→ Edited Cell→Cancel Check Out

This simplifies the cell version management since you do not need to open any additional panels. You may even assign shortcut keys for these commands.

Chapter 17:
Expert Interface Scripts (XI-Shell)

17.1: Scripting Language (LISA) for Expert Commands

EXPERT Interface (XI) shell is the extension of LISA (LANGUAGE FOR INTERFACING SIMUCAD APPLICATIONS) by commands that execute EXPERT'S actions. These commands (**xi-commands**), together with the language constructs of LISA allow you to write powerful xi-scripts for automatic generation and verification of layouts.

Some possibilities provided by the XI-SHELL are:

- saving layout into an XI-SCRIPT file;
- parameterized construction of shapes;
- P-cells (parameterized cells);
- editable activity logs with playback possibility;
- editing the layout by execution of EXPERT commands from the command line.
- automation of design tasks, like writing various logs, reading parameters from files, running verification jobs, etc.

Most xi-commands are directly related to EXPERT menu commands. For more details about these xi-commands, see the corresponding sections of this manual.

The complete description of LISA may be found in the document "LANGUAGE FOR INTERFACING SIMUCAD APPLICATIONS (LISA) USER'S MANUAL".

Please keep in mind that the set of available xi-commands constantly grows, so see the release notes for the most recent additions.

17.2: Remarks on Command Syntax and Descriptions

(1) For a general description of the syntax, see the document "LANGUAGE FOR INTERFACING SIMUCAD APPLICATIONS (LISA) USER'S MANUAL".

(2) Syntax conventions used for the xi command description:

|	Vertical bars separate possible choises of swithes for command swithes.
[ ]	Brackets define optional arguments.
>	Angle means EXPERT command line prompt.

(3) Description of xi command has the folowing sections (although, not every one is required).

Format	Format requirement for commands.
Arguments	Description of positional arguments.
Named Arguments	Description of named arguments.
Switches	Description of switches (predefined named arguments).
Return Value	Description of return value by the commands.
Examples	Actual XI code using this command.

(4) Description of xi procedure has the folowing sections (although, not every one is required).

Syntax	Syntax requirement for procedure.
Parameters	Description of parameters.
Return Value	Description of return value by the procedure.
Examples	Actual XI code using this procedure.

(5) In command arguments or procedure parameters:

- complex cell names, e.g., "`library_name::cell_name`" must always be in quotes

```
>cell open "gates013::nand2x4"
```

- directory names, and file names with the path, must always be in quotes, and the path separator must be the [/] slash or [\\] double backslash.

```
>current directory "d:/tests/GDSII"
```

(6) Command names may be truncated, as long as commands remain uniquely identified.

```
>view mode /off
>vi mo /of
```

Nevertheless, **in xi-scripts it is recommended to write full command names** for the following reason. An abbreviation for a xi-command (or LISA command) may be used as long as it is unique. However, the set of xi-commands grows together with EXPERT's capabilities. Therefore, some abbreviations may no longer be unique, and some older xi-scripts will fail to run with newer versions of EXPERT.

(7) As described below, the phrase “the command fails” means that the command will not be executed, and the execution of the script in most cases will be terminated.

(8) If a command parameter is described as `cell_name`, then whenever applicable, this name will contain the logical library name, e.g.: `“gates::inverter”` means inverter cell from gates library.

(9) Some XI commands and XI types is used extended functionality of predefined LISA data type ATTRIBUTE_ASET. There are LISA commands `'ASET_CREATE_CS'` and `'ASET_CREATE_PREDICTED_CS'` to create case-sensitive attribute sets. Their syntax is the same as for LISA commands `'ASET_CREATE'` and `'ASET_CREATE_PREDICTED'`.
A new component `'IS_CS'` for the ATTRIBUTE_SET type returns TRUE if attribute set is case sensitive, and FALSE otherwise.
A new function `'ATTR_CREATE_CS'` allows you to create a case sensitive attribute.
Notice that `'ATTR_CREATE'` function converts the attribute name to upper case to produce case-insensitive attribute.
A case sensitive attribute may be added may only to case sensitive attribute set, while a case insensitive one may be added to both types of attribute sets.

17.3: Project Commands and Procedures

17.3.1: Project Commands

PROJECT NEW

Creates a new layout project.

Format

```
PROJECT NEW proj_name [tech_name]
```

Arguments

```
proj_name (type STRING)
```

The name of the new project.

```
[tech_name]  (type STRING)
```

The file name of the technology file (optional argument).

Example 1

```
>project new “d:/projects/proj_1” “d:/tech/mux4.tcn”
```

New project with selected file name and technology is created.

Example 2

```
>project new “d:/projects/proj_2.sld” “d:/tech/05.tcn”
```

New SLD project with selected file name and technology is created.

PROJECT LOAD

Opens an existing project from disk file.

Format

```
PROJECT LOAD proj_name [tech_name] [/outputname=output_name] [/readonly]
```

Arguments

```
proj_name (type STRING)
```

The path name of the file containing the project in ELD, SLD, GDSII, CIF or Applicon format. Project type is defined by the path name extension. Extension for EXPERT'S project file is `*.eld`. You can omit path name extension for ELD project file. Allowable extensions for GDSII, CIF, and Applicon files are defined by **Setup→GDSII Input/Output**, **Setup→CIF Input/Output**, and **Setup→Applicon Input/Output** dialog panels.

```
[tech_name] (type STRING)
```

The name of a technology file (optional argument). ELD (SLD) project files store technology data inside themselves. To load an ELD (SLD) project file with external technology select the option **Use external technology file** in the **Setup→Technology→General** dialog panel.

Named Arguments

```
[/outputname=output_name] (type STRING)
```

The name of an ELD (SLD) project file for conversion from foreign format (GDSII, CIF, or Applicon).

Switches

```
[/readonly]
```

If specified a project will be opened in read-only mode. Switch is valid only for EXPERT database disk files (with extension `*.eld` or `*.sld`)

Example 1

```
>project load "d:/GDSII/mux4.GDSII" "d:/GDSII/mux4.tcn" /
outputname="mux4_org"
```

GDSII project file `"d:/GDSII/mux4.GDSII"` with technology `"d:/GDSII/mux4.tcn"` will be converted to ELD project file`"mux4_org.eld"` and the last one will be opened.

Example 2

```
>project load "d:/GDSII/mux4.gds" "d:/GDSII/mux4.tcn" /outputname="mux4.sld"
```

GDSII project file `"d:/GDSII/mux4.gds"` with technology `"d:/GDSII/mux4.tcn"` will be converted to SLD project file`"mux4_org.eld"` and the last one will be opened.

PROJECT SAVE

Saves the current project on disk in ELD, SLD or foreign formats.

Format

```
PROJECT SAVE [proj_name]
```

Arguments

```
[proj_name] (type STRING)
```

The name of the project file of type ELD, SLD, GDSII, CIF or APL according to the extension of the path name argument. The command corresponds to **File→Save** menu command if project name is not specified, and corresponds to **File→Save As...** menu command otherwise.

Examples 1

```
>project save "d:/CIF/new.cif"
```

Current project is saved as CIF project file `"d:/CIF/new.cif"`.

Examples 2

```
>project save
```

Current project is saved in the corresponding disk file.

PROJECT CLOSE

Closes the active project.

Format

```
PROJECT CLOSE [/nosave]
```

Switches

```
[/nosave]
```

The `/nosave` switch means closing a project without checking for unsaved changes in the active project.

Example

```
>project close /nosave
```

TECHNOLOGY SAVE

Saves technology on disk in a file of the type `*.tcn`.

Format

```
TECHNOLOGY SAVE tech_pathname
```

Arguments

```
tech_name (type STRING)
```

The name of technology file, with or without the extension `*.tcn`.

Example

```
>save technology "d:/GDSII/tsmc018_r3"
```

Current technology date is saved to the disk file `"d:/GDSII/tsmc018_r3"`.

17.3.2: Project Procedures

GET_LIBRARY_CELL_COUNT

Returns the number of cells in the specified library.

Syntax

```
num = GET_LIBRARY_CELL_COUNT(lib_logname)
```

Parameters

```
lib_logname (type STRING)
```

The logical name of the library, or empty string, if the cell list for the current project is requested (see example).

Return Value

```
num (type INTEGER)
```

Number of cells in the specified library.

Example

```
>num = get_library_cell_count("")
```

GET_LIBRARY_CELL_LIST

Gets the sequence of cell names in the specified library.

Syntax

```
cells = GET_LIBRARY_CELL_LIST(lib_logname)
```

Parameters

```
lib_logname (type STRING)
```

The logical name of the library, or empty string, if the cell list for the current project is requested (see example).

Return Value

```
cells (type SEQUENCE)
```

Sequence of cell names in the specified library.

Example

```
>cells = get_library_cell_list("")
```

Gets the sequence of the cell names of the current project (not including cells instanced from libraries).

GET_UNRESOLVED_CELL_LIST

Gets the sequence of cell names for the library with specified logical name.

Syntax

```
cells = GET_INRESOLVED_CELL_LIST(lib_logname)
```

Parameters

```
lib_logname (type STRING)
```

The logical name of the library, or empty string, if the cell list for the current project is requested (see example).

Return Value

```
cells (type SEQUENCE)
```

Sequence of cell names in the specified library.

Example

```
>cells = get_unresolved_cell_list("mux4")
```

Gets the sequence of the unresolved cell names at the project with logical name `'mux4'`.

GET_LIBRARY_DIRNAME

Returns the name of directory where the library with specified logical name is located.

Syntax

```
lib_dirname = GET_LIBRARY_DIRNAME(lib_logname)
```

Parameters

```
lib_logname (type STRING)
```

The logical name of the library. If empty string ""is specified then directory name of current project is returned.

Return Value

The name of directory where project with specified logical name is located (type `STRING`). Returns NIL if library with specified logical name does not exist.

Example 1

```
>display(get_library_dirname(“”))
>d:\projects
```

Disk file with current project is located in the directory “d:/projects”.

GET_LIBRARY_LOGNAME

Returns logical name of the project by pathname.

Syntax

```
lib_logname = GET_LIBRARY_LOGNAME(lib_pathname)
```

Parameters

```
lib_pathname (type STRING)
```

The name of the project file.

Return Value

```
lib_logname (type STRING)
```

The logical name of specified project file. Returns NIL if project does not have assigned logical name.

Example

```
>display(get_library_logname(“d:/mux4.eld“))
>nil
```

Project “d:/mux4.eld” does not have assigned logical name.

GET_LIBRARY_PATHNAME

Returns path name of the library by logical name.

Syntax

```
lib_pathname = GET_LIBRARY_PATHNAME(lib_logname)
```

Parameters

```
lib_logname (type STRING)
```

The logical name of the library. If empty string ““is specified then path name of current project is returned.

Return Value

Full pathname of specified library (type STRING). Returns NIL if library with specified logical name does not exist.

Example 1

```
>display(get_library_pathname(“”))
>d:/mux4.eld
```

“d:/mux4.eld” is a disk file name of current project.

Example 2

```
>display(get_library_pathname(“A“))
>nil
```

Library with logical name “A” does not exist.

17.4: Cell Commands and Procedures

17.4.1: Cell Commands

CELL CLOSE

Closes the current or all opened cell windows.

Format

```
CELL CLOSE [/current | /all]
```

Switches

```
/current | /all]
```

- `/current` (default switch) closes the current cell window.
- `/all` closes all opened cell windows.

Example

```
>cell close;
```

Current cell window is closed.

CELL CREATE

Creates a new cell from the set of selected objects.

Format

```
CELL CREATE cell_name [/inplace]
```

Arguments

```
cell_name (type STRING)
```

The name of a new cell.

Switches

```
[/inplace]
```

If specified, then selected objects of the edited cell will be replaced by the instance of the new cell.

Example 1

```
>cell create "cell_new"
```

Creates a new cell "`cell_new`" from selected objects.

Example 2

```
>cell create "cell_new" /inplace
```

Replaces the selected objects by the instance of the new cell "`cell_new`" created from them.

CELL COPY

Creates a copy of specified cells in a target library.

Format

```
CELL COPY {cell_name} [target_lib_logname]
[/sourcemap=source_cell_names]
[/targetmap=target_cell_names]
[/hierarchy]
[/nolibcells | /libcells = (true, false)]
```

Arguments

```
{cell_name} (type SEQUENCE)
{cell_name}={cell_name1, …, cell_namen} is a sequence of cells (represented
by names) to be copied.
[target_lib_logname] (type STRING)
```

Logical name of target library. If library logical name is not specified, then cells are copied to the current project.

Named Arguments

```
[/sourcemap = source_cell_names] (type ATTRIBUTE_SET)
```

Case sensitive (EXPERT's cell names are case sensitive) attribute sets are used to set new names for copied cells. It is needed, for example, to resolve collision between names of copied cells and cells existing in the target project. The name of each attribute in the `source_cell_names` is a name of copied cell (type `STRING`). The attributed value is a name of this cell in the target project (type `STRING`).

Note: Full cell name (included logical name of library) should be used for the original cell, and only the cell name (without the logical name of the library) should be used for a new cell name in the target project.

```
[/targetmap = target_cell_names] (type ATTRIBUTE_SET)
```

Case sensitive (EXPERT's cell names are case sensitive) attribute sets are used to set new names for the cells in the target project. It is needed, for example, to resolve collision between names of copied cells and cells existing in the target project. The name of each attribute in the `target_cell_names` is a name of an existing cell (type `STRING`). The attributed value is a new name of this cell in the target project (type `STRING`).

Note: Cell names without a library logical name should be used for both cell names.

Named Argument

```
[/nolibcells | /libcells = ] (type BOOLEAN)
```

If `/libcells` is specified or assigned a value of true, then `CELL COPY` with a hierarchy will be performed with the whole cell hierarchy, otherwise (`/nolibcells` or `/libcells = false`) cells from linked libraries will be ignored.

Switches

```
[/hierarchy]
```

If the switch `/hierarchy` is specified, then cells will be copied with hierarchy.

Example

```
>aCells = {“ad5::CBBASE”, “05::CBBASE”, “ad5::PLYCNR”, “05::PLYCNR”, -
>”ad5::VCR”, “05::VCR”, “05::VCCTB”};
>mapSourceCells = aset_create_cs(aCells.size);
>mapSourceCells[“05::CBBASE”] = “CBBASE_05”;
>mapSourceCells[“05::PLYCNR”] = “PLYCNR_05”;
>mapSourceCells[“05::VCR”] = “VCR_05”;
>cell copy (aCells) /sourcemap = (mapSourceCells) /hierarchy;
```

CELL DELETE

Deletes the set of cells given by the list of cell names from the project. If a cell is a read-only cell, or its parent library is read-only, then it is not deleted.

Format

```
CELL DELETE {cell_name}
```

Arguments

```
{cell_name} (type SEQUENCE)
```

`{cell_name} = {cell_name1, …, cell_namen}` is a sequence of cells (represented by names) to be deleted.

Switches

```
 [/hierarchy]
```

If the switch `/hierarchy` is specified, then all cells which have become top cells, as a result of delete operation, and will be deleted.

Example

```
>cell delete {“inverter1”, “nand2”}
```

Cells `“invert”` and `“nand2”` is deleted from current project.

CELL DISCARD

Discards all changes in the specified cell.

Format

```
CELL DISCARD [cell_name]
```

Arguments

```
[cell_name] (type STRING)
```

Specifies the name of the cell whose changes will be discarded. If the parameter is not set, the current active cell is affected.

Example

```
>cell discard “inverter”;
```

CELL EXPORT

Exports cells into the ELD, XI-script, CIF, Applicon, or GDSII format on disk.

Format

```
CELL EXPORT {cell_name} file_pathname [/multifile]
[/nolibcells | /libcells = (true, false)]
```

Arguments

```
{cell_name} (type SEQUENCE)
```

`{cell_name} = {cell_name1, …, cell_namen}` is a sequence of cells (represented by names) to be exported.

```
file_pathname (type STRING)
```

The path name of an output file. The format ELD, XI-script, CIF, Applicon, or GDSII is defined according to the extension of the `file_pathname`.

Named Argument

```
[/nolibcells | /libcells = ] (type BOOLEAN)
```

If `/libcells` is specified or assigned a value of true, then `CELL EXPORT` with a hierarchy will be performed with the whole cell hierarchy, otherwise (`/nolibcells` or `/libcells = false`) cells from linked libraries will be ignored.

Switches

```
[/multifile]
```

The switch `/multifile` means that specified cells will be exported to separate disk files. This switch is applied for export cells to the ELD or GDSII formats only.

Examples

```
>cell export “inverter”,”nand2” “mux1.gds” /multifile
>cell export “inverter”,”nand2” “mux1.gds”
>cell export “inverter” “mux1.cif”
>cell export “inverter” “mux1.xis”
```

CELL IMPORT

Adds cells from another project (stored in ELD, GDSII or Applicon formats) to the current project.

Format

```
CELL IMPORT project_pathname [{cell_name}]
[/sourcemap=source_cell_names]
[/targetmap=target_cell_names]
[/nolibcells | /libcells = (true, false)]
```

Arguments

```
[project_pathname] (type STRING)
```

The path name of a project disk file containing the required cells.

```
{cell_name} (type SEQUENCE)
```

`{cell_name} = {cell_name1, …, cell_namen}` is a sequence of cells (represented by names) imported from specified project to the current project. If this parameter is not set, then all cells from file project_pathname will be imported.

Named Arguments

Note: Cell names without a library logical name should be used in all named arguments for this command.

```
[/sourcemap = source_cell_names] (type ATTRIBUTE_SET)
```

Case sensitive (EXPERT's cell names are case sensitive) attribute sets are used to set new names for imported cells. It is needed, for example, to resolve collision between names of copied cells and cells existing in the current project. The name of each attribute in the `source_cell_names` is the name of an imported cell (type `STRING`). The attribute value is the name of this cell in the current project (type `STRING`).

```
[/targetmap = target_cell_names] (type ATTRIBUTE_SET)
```

Case sensitive (EXPERT's cell names are case sensitive) attribute sets are used to set new names for the cells in the current project. It is needed, for example, to resolve collision between names of copied cells and cells existing in the current project. The name of each attribute in the `target_cell_names` is the

name of an existing cell (type STRING). The attribute value is the new name of this cell in the current project (type STRING).

Example

```
>aCells = {"inverter", "mux1"};
>mapTargetCells = aset_create_cs();
>mapTargetCells["mux1"] = "mux1_old1"
>mapTargetCells["nand2"] = "nand2_old1";
>mapTargetCells["inverter"] = "inverter_old1";
>cell import "mux4.gds" (aCells) /targetmap = (mapTargetCells);
```

Cells "inverter" and "mux1" will be imported from project "mux4.gds". Cell "nand2" from a specified project will also be copied because there are instances if "nand2" in the cell "mux1". Note that the cell import operation is always hierarchical. Cells existing in the current project will be renamed according to named argument mapTargetCells.

Named Argument

```
[/nolibcells | /libcells = ] (type BOOLEAN)
```

If /libcells specified or assigned a value of true, then CELL IMPORT will be performed with the whole cell hierarchy, otherwise (/nolibcells or /libcells = false) cells from linked libraries will be ignored.

CELL OPEN

Opens cell view for the specified cell.

Format

```
CELL OPEN [cell_name]
```

Arguments

```
cell_name (type STRING)
```

The name of the required cell. If the cell does not exist, then a new cell is created and added to the project. The CELL OPEN command without parameters displays the name of the current cell.

Examples 1

```
>cell open "gates::inverter"
```

Opens cell "inverter" from the library "gates".

Examples 2

```
>cell open "inverter"
```

Opens cell view for the cell "inverter" from the current project.

CELL NEW

Creates a new cell in the project.

Format

```
CELL NEW cell_name [/pcell = (true | false)]
[/textual | /graphical]
[/parnames =({pname_1,…,pname_n})]
[/params =({pvalue_1,…,pvalue_n})]
[/params =({pvalue_1,…,pvalue_n})]
```

```
[/pattern_cell = cell_name] (Type String)
```

Arguments

```
cell_name (type STRING)
```

The name of the new cell.

Named Arguments

```
[/pcell = (true | false)] (type BOOLEAN)
```

If specified or assigned a value of true, the new P-cell will be added to the current cell library.

```
[/textual | /graphical]
```

- `/textual` means than textual P-cell will be created.
- `/graphical` is for graphical P-cell creation.

The argument cluster is not used for regular cells.

```
[/parnames = ({pname_1, …, pname_n})]
```

Sets the P-cell parameter names.

```
[/params =({pvalue_1, …, pvalue_n})]
```

Sets the P-cell parameter values.

```
[/pattern_cell = cell_name] (Type String)
```

Specifies the cell by its name. The specified cell is used for creating initial content of the new P-cell: P-cell Xi script in the case of textual P-cell, and copy of specified cell geometry when graphical P-cell is created.

Example 1

```
>cell new “inverter1”
```

Creates cell “`inverter1`” and adds it to the current project.

Example 2

```
>cell new “mux::inverter”
```

Creates cell “`inverter1`” in the library with logical name “`mux`”.

CELL PRINT

Plots the visible part of the layouts using the current plotting parameters. Plotting parameters can be exported to the EXPERT settings file (**Setup→Export Setup...**). Using XI command `IMPORT EXPERT SETTINGS` you may print the current layout with particular plotting parameters

Format

```
CELL PRINT
```

Example

```
>cell print;
```

CELL RENAME

Renames the specified cell.

Format

```
CELL RENAME old_cell_name new_cell_name
```

Arguments

```
old_cell_name (type STRING)
```

The name of cell.

```
new_cell_name (type STRING)
```

The new name of cell.

Example

```
>rename cell "inverter" "inv1"
```

CELL SAVE

Saves the listed cells.

Format

```
CELL SAVE [{cell_name}]
```

Arguments

```
{cell_name} (type SEQUENCE)
```

`{cell_name} = {cell_name1, …, cell_namen}` is a sequence of cells to be saved. If the command has no arguments, then the current cell is saved.

Example

```
> cell save {"inverter", "nand2"};
```

Cells "inverter" and "nand2" is saved.

CELL SAVEAS

Creates a copy of the cell, and adds the new copied cell to the project. Discards all changes in the source cell. Used to save last modifications of the specified cell to a new one and restore the original content.

Format

```
CELL SAVEAS cell_name new_cell_name
```

Arguments

```
cell_name (type STRING)
```

The name of the cell for copying operation.

```
new_cell_name (type STRING)
```

The name of the new cell.

Example

```
> cell saveas "inverter", "inv_1";
```

CELL STATUS

Sets or clears the "read-only" cell status.

Format

```
CELL STATUS cell_name [/r | /rw]
```

Arguments

```
cell_name (type STRING)
```

The name of the cell. If the argument is not specified, then status will be set for the current cell.

Switches

```
[/r | /rw]
```

The /r switch sets “read only” cell status, and the /rw switch removes it.

Example

```
> cell status “inverter” /r;
```

Sets “read only” status for cell inverter.

DEFINE PCELL

Defines a new or redefines the existing EXPERT'S PCell.

Format

```
DEFINE PCELL pcell_name [/REPLACE]
  PARAMETER par1_name
  ...
  PARAMETER parN_name
  BODY pcell_body;
```

Arguments

pcell_name (type STRING) is the name of the PCell. It must contain the library logical name (full cell name) if applicable.

Switches

```
[/REPLACE]
```

Specifies that the previous declaration of the PCell with this name will be superseded.

Subcommands

```
PARAMETER
```

Describes the parameters for the PCell

Format

```
PARAMETER parameter_name /TYPE = (type) /DEFAULT = (value)
```

Arguments

parameter_name (type STRING) is the name of the PCell parameter.

Named Arguments

```
/TYPE = (type)
```

Specifies the valid type that can be entered for PCell parameter: INTEGER, DOUBLE, STRING, BOOLEAN.

```
/DEFAULT = (value)
```

Specified default value for PCell parameter

```
BODY
```

Provides the implementation of EXPERT'S PCell.

Format

```
DO body
```

Arguments

```
body
```

A block that defines the PCell's implementation.

Example

```
DEFINE PCELL "square" /REPLACE
  PARAMETER posx /TYPE=(DOUBLE) /DEFAULT=(0.0)
  PARAMETER posy /TYPE=(DOUBLE) /DEFAULT=(0.0)
  PARAMETER size /TYPE=(DOUBLE) /DEFAULT=(5.0)
  BODY BEGIN
  box (posx) (posy) (size) (size);
  box (posx) (posy) (size * 2) (size * 2);
END;
```

17.4.2: Cell Procedures

CELL_EXISTS

Returns TRUE if specified cell exists, and FALSE if otherwise.

Syntax

```
ret = CELL_EXISTS (cell_name)
```

Parameters

`cell_name (type STRING)`

The name of the cell.

Return Value

`ret (type BOOLEAN)`

TRUE if the cell exists, and FALSE if otherwise.

Example

```
display(cell_exists(“nand2”));
```

GET_CELL_BBOX

Calculates the bounding box of the cell.

Syntax

```
bbox = GET_CELL_BBOX(cell_name)
```

Parameter

`cell_name (type STRING)`

The name of the cell.

Return Value

`bbox (type RECT)`

The bounding box of the cell with the name `cell_name`.

Example

```
>bb = get_cell_bbox(“gates::NAND2”)
```

Calculates the bounding box of the cell `“NAND2”` from the library with the logical name `“gates”` and places it into the variable `‘bb’`, where

- `b.ypos` is the y-coordinate of the lower side of the bounding box;
- `bb.xpos` is the x-coordinate of the left side of the bounding box;
- `bb.ypos+box.ysize` is the y-coordinate of the top side of the bounding box.
- `bb.xpos+box.xsize` is the x-coordinate of the right side of the bounding box.

GET_CELL_OBJECT_COUNT

Returns the number of layout objects in the specified cell.

Syntax

```
count = GET_CELL_OBJECT_COUNT(cell_name)
```

Parameters

```
cell_name (type STRING)
```

The name of the cell.

Return Value

```
count (type INTEGER)
```

Number of layout objects in the cell `cell_name`.

Example

```
>num = get_cell_object_count(“gates::INVERTER”)
```

GET_CELL_STATUS

Returns TRUE if the specified status is set, and FALSE is otherwise.

Syntax

```
ret = GET_CELL_STATUS(cell_name, status_flag)
```

Parameters

```
cell_name (type STRING)
```

The name of the cell.

```
status_flag
```

`status_flag` can be one of following predefined constants:

- `CS_TOP`: the cell is a top-level cell in the project’s hierarchy.
- `CS_READONLY`: the cell is read only (e.g., not checked out).
- `CS_CHECKEDOUT`: the cell is from the shared library and checked out.
- `CS_MODIFIED`: the cell is modified, but not saved.
- `CS_PARAMETERIZED`: the cell is a P-cell.

Return Value

```
ret (type BOOLEAN)
```

TRUE if the status is set, and FALSE otherwise.

Example

```
>display(get_cell_status(“nand2”, CS_TOP));
```

GET_EDITED_CELL_NAME

Returns the full name of the edited cell.

Syntax

```
cell_name = GET_EDITED_CELL_NAME()
```

Return Value

```
cell_name (type STRING)
```

The name of the edited cell.

Example

```
>cell_name = get_edited_cell_name()
```

GET_PCELL_PARAMETER_NAMES

Returns sequence of parameter names for specified cell.

Syntax

```
param_names = GET_PCELL_PARAMETER_NAMES(cell_name)
```

Parameters

```
cell_name (type STRING)
```

The name of the parameterized cell.

Return Value

```
param_names (type SEQUENCE)
```

Sequence of parameter names for a specified parameterized cell. If the cell does not exist, or is not parameterized, then NIL value is returned.

Example

```
>param_names = get_pcell_parameter_names("INV")
```

GET_PCELL_PARAMETER_DEFAULTS

Returns a sequence of default parameter values for a specified cell.

Syntax

```
param_defs = GET_PCELL_PARAMETER_DEFAULTS(cell_name)
```

Parameters

```
cell_name (type STRING)
```

The name of the parameterized cell.

Return Value

```
param_defs (type SEQUENCE)
```

Sequence of default parameter values for a specified parameterized cell. If the cell does not exist, or is not parameterized, then NIL value is returned.

Example

```
>param_defs = get_pcell_parameter_defaults("INV")
```

GET_SELECTED_OBJECT_COUNT

Returns the number of selected objects in the specified cell.

Syntax

```
num = GET_CELL_OBJECT_COUNT(cell_name)
```

Parameters

```
cell_name (type STRING)
```

The name of the cell.

Return Value

```
num (type INTEGER)
```

The number of selected objects in the cell `cell_name`.

Example

```
>num = get_selected_object_count("gates::INVERTER")
```

17.5: Cell View Commands and Procedures

17.5.1: Cell View Commands

CELL OPEN

See "CELL OPEN" on page 17-12.

CELLVIEW REDRAW

Refreshes the screen image in the specified region of the layout in the current window.

Format

```
CELLVIEW REDRAW [domain]
```

Arguments

```
domain (Type RECT)
```

Specifies the region to be redrawn. If the argument is not specified, the whole image of the current window will be refreshed.

Example 1

```
>dom = rect_create(0, 0, 3, 4)
>cellview redraw (dom)
```

REFERENCE POINT

Sets a temporary reference point to be used by the command, and activates or deactivates it. `REFERENCE POINT` command without parameters displays the current reference point.

Format

```
REFERENCE POINT [x] [y] [/activate | /deactivate]
```

Arguments

```
x (Type INTEGER, FLOAT, DOUBLE)
y (Type INTEGER, FLOAT, DOUBLE)
```

(`x`, `y`) are the absolute coordinates of the reference point in the user units.

Switches

```
[/activate | /deactivate]
```

Example 1

```
>reference point 20 30
```

```
>reference point
>Reference Point: RX = 0 RY = 0
```

Sets the reference point (20, 30) but does not activate it.

Example 2

```
>reference point 20 30 /activate
>reference point
>Reference Point: RX = 20 RY = 30
```

Sets and activates the reference point (20, 30).

STATUSBAR

Updates the message in the EXPERT Status Bar.

Format

```
STATUSBAR msg
```

Arguments

```
msg (Type STRING)
```

Specifies the text to show in the EXPERT Status Bar

ZOOM

Expands or reduces the view of the current cell.

Format

ZOOM [x] [y] [x_s] [y_s] [/all /in | /out | /selected | /window]

Arguments

x (Type INTEGER, FLOAT, DOUBLE)
y (Type INTEGER, FLOAT, DOUBLE)
x_s (Type INTEGER, FLOAT, DOUBLE)
y_s (Type INTEGER, FLOAT, DOUBLE)

(x, y) defines the left bottom point of window box for zooming to window. x_s and y_s define dimension of the window box. Note, arguments are valid if the switch /window is selected.

Switches

[/all | /in | /out | /selected | /window]

- /all shows the whole layout of the active cell.
- /in enlarges the content of the layout window for a more detailed view.
- /out decreases the resolution of the image allowing a more detailed overview.
- /selected allows you to investigate selected objects.
- /window shows part of the active cell located inside bounding box defined by arguments x, y, x_s, y_s.

Examples

```
>zoom /all
>zoom /w 0 0 10 10
```

17.5.2: Cell View Procedures

CELLVIEW_GET_VIEW_RECT

Returns the rectangular area of the current cell view window in the user units.

Syntax

```
view_rect = CELLVIEW_GET_VIEW_RECT()
```

Parameters

None

Return Value

```
view_rect (type RECT).
```

View a rectangle of the current cell view. The components of the rectangle are in the current measurement units.

Example

```
>cur_view_rec = cellview_get_view_rect()
```

17.6: Setup Commands

17.6.1: Set Directories Commands

CURRENT DIRECTORY

Sets the default directory for project open/save operations, both for file browser and for execution of xi-script for files which are specified without a path name in xi-commands

Format

```
CURRENT DIRECTORY dir_name
```

Arguments

```
dir_name (type STRING)
```

Name of directory.

Example 1

```
>current directory “f:/data”
>project load “mux4.eld”
```

Project file “f:/data/mux4.eld” will be loaded.

Example 2

```
>current directory “C:/WinNT”
>spawn “notepad.exe“
```

The “Notepad” Windows application will be started.

TECHNOLOGY DIRECTORY

Sets the default directory where technology files are located, both for the file browser and for execution of xi-commands that specify a technology file without a full path.

Format

```
TECHNOLOGY DIRECTORY dir_name
```

Arguments

```
dir_name (type STRING)
```

Name of directory.

Example 1

```
>current directory "f:/data
>technology directory "f:/tcn03"
>project load "mux4.gds" "mux4.tcn"
```

The GDSII file "f:/data/mux4.gds" using the technology file "f:/tcn03/mux4.tcn" will be loaded.

17.6.2: Expert Settings Commands

EXPORT EXPERT SETTINGS

Saves EXPERT settings into a file (with the extension *.ecf) for latter use.

Format

```
EXPORT EXPERT SETTINGS path_name
```

Arguments

```
path_name (type STRING)
```

Path name of EXPERT settings file.

Example

```
>export expert settings "d:/data/expert.ecf";
```

IMPORT EXPERT SETTINGS

Loads EXPERT settings from file with the extension *.ecf.

Format

```
IMPORT EXPERT SETTINGS path_name
```

Arguments

```
path_name (type STRING)
```

Path name of EXPERT'S settings file.

Example

```
>import expert settings "d:/data/expert.ecf";
```

RESTORE EXPERT SETTINGS

Restores EXPERT settings, either default ones, or from the registry of the current user.

Format

```
RESTORE EXPERT SETTINGS [/default | /registry]
```

Switches

```
[/default | /registry]
```

If /default *is* specified, then predefined EXPERT settings will be applied. If /registry switch or no switch is specified, the setting will be read from the registry.

Example 1

```
>restore expert settings /default;
```

Predefined EXPERT settings will be applied

Example 2

```
>restore expert settings;
```

EXPERT settings stored in the registry will be applied.

SAVE EXPERT SETTINGS

Writes EXPERT settings into the registry of the current user.

Format

SAVE EXPERT SETTINGS

Arguments

None

Example

```
>save expert settings;
```

17.6.3: Expert Setup Commands

SETUP GRID

Sets parameters for the common, layer-specific, or cell instance grid (similarly to **Setup→Technology→Grid** dialog panel).

Format

```
SETUP GRID /commongrid | /layergrid | /instancegrid
[/layer=layername]
[/minstep=value]
[/equalxy | /noequalxy]
[/snap | /nosnap]
[/snapstepx=value]
[/snapstepy=value]
[/snapdependvisible | /nosnapdependvisible]
[/snapratiox=value]
[/snapratioy=value]
[/visible | /invisible]
[/visiblestepx=value]
[/visiblestepy=value]
[/supergrid | /nosupergrid]
[/sgridstepx=value]
[/sgridstepy=value]
[/autogrid | /noautogrid]
[/autonormal | /autodense | /autosparse]
[/dottedgrid | /solidgrid]
[/color={r,g,b}]
```

Arguments

None

Named Arguments

`[/layer=layername] (type STRING)`

Specifies the layer for which grid settings will be applied (goes with `/layergrid` switch only).

`[/minstep=value] (type INTEGER)`

Specifies the minimal grid step in nanometers.

`[/snapstepx=value] (type INTEGER)`

Specifies the snap grid step in minimal grid units.

`[/snapstepy=value] (type INTEGER)`

Specifies the snap grid step in minimal grid units along Y direction if different from the snap grid along X direction.

`[/snapratiox=value] (type INTEGER)`

Specifies the ratio between visual and snap grid (now supported ratio: 1 (1:1), 2 (1:2), 5(1:5), and 10(1:10)).

`[/snapratioy=value] (type INTEGER)`

Specifies the ratio between visual and snap grid along Y direction if different from the ratio along X direction.

`[/visiblestepx=value] (type INTEGER)`

Specifies the visual grid step in minimal grid units.

`[/visiblestepy=value] (type INTEGER)`

Specifies the visual grid step in minimal grid units along Y direction if different from the visual grid step in X direction.

`[/sgridstepx=value] (type INTEGER)`

Specifies the super grid step over the visual grid.

`[/sgridstepy=value] (type INTEGER)`

Specifies the super grid step over the visual grid along Y direction if different from X direction.

`[/color={r,g,b}] (type SEQUENCE)`

Sequence of numbers between 0 and 255 that specifies the RGB value of the grid color.

Switches

`/commongrid | /layergrid | /instancegrid`

- `/commongrid` switch means that default grid settings will be modified.
- `/layergrid` switch means that grid settings will be applied to the particular layer with the name specified by argument `/layer`.
- `/instancegrid` switch means that grid settings will be applied to the cell instance/array placement.

`[/equalxy | /noequalxy]`

- `/equalxy` switch means that grid step is equal along X and Y directions.
- `/noequalxy` switch means that grid step along X direction is different from grid step along Y direction.

[/snap | /nosnap]

- /snap switch means that cursor movement will be limited by snap grid nodes controlled by named arguments /snapstepx and /snapstepy.
- /nosnap switch means that snap grid mode is OFF.

[/snapdependvisible | /nosnapdependvisible]

- /snapdependvisible switch means that snap grid size is dependable from visual grid size.
- /nosnapdependvisible switch means that snap grid size is defined by correspond named arguments /snapstepx and /snapstepy.

[/visible | /invisible]

- /visible and /invisible switches make grid nodes visible and invisible correspondingly.

[/supergrid | /nosupergrid]

- /supergrid and /nosupergrid switches turn super grid nodes ON and OFF correspondingly.

[/autogrid | /noautogrid]

/autogrid switch means that visual grid will be generated automatically.

- /noautogrid switch turns OFF automatic visual grid.

[/autonormal | /autodense | /autosparse]

- /autonormal, /autodense and /autosparse are different types of density for automatic grid.

[/dottedgrid | /solidgrid]

- /dottedgrid means that the grid will be represented by dots.
- /solidgrid means that the grid will be drawn as lines.

SETUP SHAPE

Sets parameters for approximation of ellipses, wires, donuts and rolled regions by polygonal regions.

Format

```
SETUP SHAPE
[/OffgridWire | /noOffgridWire]
[/offgridCircle | /noOffgridCircle]
[/firstascenter | /nofirstascenter]
[/circumscribed | /inscribed]
[/vertices | /angle | /deflection]
[/value = ]
```

Arguments

None

Named Arguments

[/value=] (Type DOUBLE)

Specifies the value for the specified polygonal curve approximation method. The number of vertices for polygonal approximation if the /vertices switch selected; angular distance between vertices in degrees if the /angle switch selected; deflection distance of polyline from the curve in user units if the /deflection switch selected.

Switches

[/offgridwire | /nooffgridwire]

Affects the wire drawing and "convert to polygon" operation for wire objects.

- /offgridwire switch means that the wire will be approximated by region using centerline and constant width.
- /nooffgridwire switch means that no off-grid vertices of the resulting region are allowed.

[/offgridCircle | / offgridCircle]

- /offgridCircle switch allows you to snap on the grid only circle/ellipse centers and radii.
- /nooffgridCircle means that each vertex of the resulting region will be snapped to the grid.

[/firstascenter | /nofirstascenter]

- /firstascenter switch selects first vertex of approximating polygon with the same Y coordinate as the center of circle/ellipse.
- /firstvertical switch creates polygon with first vertex on the vertical side.

[/circumscribed | /inscribed]

- /circumscribed switch specifies the approximation by circumscribed polygon.
- /inscribed switch specifies the approximation by inscribed polygon.

[/vertices | /angle | /deflection]

- /vertices switch means that approximation is defined by number of vertices (the value is defined by named argument /value).
- /angle specifies that approximation is defined by angular distance between vertices of the resulting polygon (the value is defined by named argument /value).
- /deflection switch means that maximal deflection of the polyline from curve does not exceed the prescribed value (value is defined by named argument /value).

Example 1

```
>SETUP SHAPE /offgridwire /offgridCircle
```

Sets modes for approximation of wires and circles/ellipses that allows offgrid geometry.

Example 2

```
>SETUP SHAPE /vertices /value = 20
```

Specifies that circles, ellipses, donuts and arcs of rolled regions will be approximated by polygons with 20 vertices.

UNITS

Selects the measurement unit and the value for a "Lambda" unit. Command UNITS without parameters displays the value of the current unit.

Format

```
UNITS [unit_size] [/nm | /um | /mil | /mm | /cm | /inch | /m | /lambda]
```

Arguments

```
[unit_size] (type INTEGER, FLOAT, DOUBLE)
```

Value of user-defined "Lambda" unit in microns.

Switches

```
[/nm | /um | /mil | /mm | /cm | /inch | /m | /lambda]
```

The unit of measurement.

Example 1

```
>units /lambda 2
```

One user unit equals `2 um`.

Example 2

```
>units /um
```

Microns are set for measurements

17.6.4: XI Setup Commands

NONSTOP MODE

Sets nonstop mode for XI script execution.

Format

```
NONSTOP MODE [/on | /off]
```

Switches

```
[/on | /off]
```

`/on` does not interrupt execution of the XI script on warnings/messages. If a message requires user's choice ("yes/no", etc.), the default option is assumed.

Example

```
>nonstop mode /on
```

QUIET MODE

Sets quiet mode (no warnings) for XI commands execution.

Format

```
QUIET MODE [/on | /off]
```

Switches

```
[/on | /of]
```

`/on` does not display warnings messages during execution of the XI commands; `/off` works otherwise.

Example

```
>quiet mode /on
```

VIEW MODE

Sets view mode for XI command line. If this mode is on, then the results of subsequent xi-commands executed from the EXPERT command line will be immediately displayed by EXPERT.

Format

```
VIEW MODE [/on | /off]
```

Switches

```
[/on | /off]
```

Example

```
>view mode /on
```

17.6.5: Edit Settings Commands

CELL ACTIVATE

Assigns a cell for subsequent commands for placing cell instances/arrays.

Format

```
CELL ACTIVATE [cell_name]
```

Arguments

```
cell_name (type STRING)
```

The name of the required cell. `CELL ACTIVATE` command without a parameter displays the name of the current activated cell.

Example

```
>cell activate "nand2"
>instance 0 0
```

Instance of the cell `"nand2"` will be placed at position (`0`, `0`).

CELL OPEN

See "CELL OPEN" on page 17-12.

CUT MODE

Sets a mode for `CUT LINE` command.

Format

```
CUT MODE [/selected | /active | /all]  [/savewire | /nosavewire]
```

Arguments

None

Switches

The switches correspond to menu settings **Tools→Cut by Line Mode**.

`[/selected | /active | /all]`

- `/selected` switch means that only selected objects are cut.
- `/active` switch allows to cut object lying on the active layer.
- `/all` switch means that all objects are cut.

`[/savewire | /nosavewire]`

- `/savewire` switch means that cut-by-line operation does not convert wires into polygon.
- `/nosavewire` is opposite mode to `/savewire`.

Example

```
>cut mode /all
```

LAYER

Sets the active layer.

Format

```
LAYER [layer_name]
```

Arguments

```
layer_name (type STRING)
```

The name of the layer (this layer must exist in the active project). LAYER command without the parameter displays the name of the active layer.

Example

```
>layer “Metal2”
>box 0 0 4 5
```

box is created on the active layer “Metal2”.

REGION MODE

Sets a mode for object creation according to the setting from **Edit→Region Modes** submenu**.**

Format

```
REGION MODE [/normal | /merge | /hole | /holeall | /holeselected]
```

Switches

```
[/normal | /merge | /hole | /holeall | /holeselected]
```

- /normal switch is used to create a new shape.
- /merge switch means that new object will be merged with existing ones.
- /hole switch means that new object will be treated as hole in the existing ones in the active layer.
- /holeall works like /hole but hole is created in the overlapping objects on the all layers.
- /holeselected switch means that hole will be created in the overlapping selected objects.

Example

```
>box 0 0 3 3;
>region mode /merge
>box 0 3 3 3;
```

Region with bounding box (0,0, 3,6) will be created.

SELECT MODE

Defines selection settings.

Format

```
SELECT MODE
[/noinstance | /instance]
[/noprimitive | /primitive]
[/reselect | /select | /deselect]
[/cross | /inside]
[/boundary | /solid]
```

Switches

```
[/noinstance | /instance]
```

- /noinstance switch means that instances will be ignored during selection.
- /instance switch means that instances will not be ignored during selection.

```
[/noprimitive | /primitive]
```

- /noprimitive switch means that primitives will be ignored during selection.
- /primitive switch means that primitives will not be ignored during selection.

[/reselect | /select | /deselect]

- /reselect switch means that all previously selected objects will be deselected before select operation.
- /select switch means that new objects will be selected without change of selection state of other objects.
- /deselect switch means that objects will be deselected.

[/cross | /inside]

- /cross mode means that all object intersected by selection box will be affected by the selection command.
- /inside mode means that objects that are completely inside of the selection box will be affected by the selection command.

[/boundary | /solid]

- /boundary switch means that objects will be selected by point at the border line.
- /solid switch mean that objects will be selected by any point inside object.

Example

```
>select mode /reselect /inside
```

17.6.6: Getting/Releasing Simucad Licenses Commands

When DRC or netlist extraction is run from within EXPERT, the corresponding license is taken/released for each run of DRC/extraction. In some cases, such as when running big batches of jobs for small cells, license search may slow down the execution, or even interrupt it if someone grabs the last license between two DRC runs of a batch job

LICENSE GET

Gets Simucad licence.

Format

```
LICENCE GET /GUARDIAN_DRC | /GUARDIAN_LPE
```

Switches

/GUARDIAN_DRC | /GUARDIAN_LPE

- /GUARDIAN_DRC switch means that DRC license will be taken.
- /GUARDIAN_LPE switch means that EXTRACTOR'S license will be taken.

Example

```
> LICENSE GET /GUARDIAN_DRC
```

DRC license is taken.

LICENSE RELEASE

Gets Simucad licence.

Format

```
LICENCE RELEASE  /GUARDIAN_DRC | /GUARDIAN_LPE
```

Switches

```
/GUARDIAN_DRC | /GUARDIAN_LPE
```

- `/GUARDIAN_DRC` switch means that DRC license will be taken.
- `/GUARDIAN_LPE` switch means that Extractor's license will be taken.

Example

```
> LICENSE RELEASE /GUARDIAN_LPE
```

Netlist extraction license is released.

17.7: Layer Commands and Procedures

17.7.1: Layer Commands

DERIVATION SCRIPT

Stores the DRC script for the all derived layers of current technology into the specified disk file.

Format

```
DERIVATION SCRIPT path_name
```

Arguments

```
path_name (type STRING)
```

Specifies the pathname where full derivation script is stored.

DERIVED CLEAN

Cleans derived layers in the specified cell scope.

Format

```
DERIVED CLEAN [/primitives | /hierarchy | /project]
```

Switches

```
[/primitives | /hierarchy | /project]
```

Argument cluster defines the scope where the derived layers will be cleaned up.

- `/primitive` means that primitive layout objects laying on the derived layers of the current opened cell will be deleted.
- `/hierarchy` cleans derived layers in the current cell, and derived layers of its whole cell hierarchy.
- `/project` cleans derived layers in the all cells of the loaded cell library.

DERIVED GENERATE

Generates derived layers.

Format

```
DERIVED GENERATE [/active | /all] [/primitives | /cellwise | /flat | /layer]
```

Switches

```
[/active | /all]
```

- `/active` (default switch) means that the active derived layer will be generated.
- `/all` invokes derivation of all derived layers described in the current technology.

DERIVED LAYER

Prevents recalculating derived layers after the last one is activated.

Format

```
DERIVED LAYER [/freeze = (true | false)]
```

Argument

```
[/freeze = (true | false)] (type BOOLEAN)
```

If specified, or `/freeze = true` then the derived layer is not calculated upon activation; otherwise the derived layer will be recalculated.

LAYERPLAN LOAD

Loads layer plans from a specified disk file.

Note: Layer plan should be compatible with current technology.

Format

```
LAYERPLAN LOAD pathname
```

Arguments

```
pathname (Type STRING)
```

Specifies the name of a disk file that contains EXPERT layer plans.

Example

```
>layerplan load "d:\\temp\\asa.elp"
```

SCRATCH CLEAN

Cleans the scratch layers in the specified cell scope.

Format

```
SCRATCH CLEAN [/primitives | /hierarchy | /project]
```

Switches

```
[/primitives | /hierarchy | /project]
```

Argument cluster defines the scope where the scratch layers will be cleaned up.

- `/primitive` means that primitive layout objects laying on the scratch layers of the current opened cell will be deleted.
- `/hierarchy` cleans scratch layers in the current cell and scratch layers of its whole cell hierarchy.
- `/project` cleans scratch layers in the all cells of the loaded cell library.

SCRATCH DELETE

Removes the scratch layer definitions from technology.

Format

```
CRATCH DELETE
```

Arguments

None.

NEW LAYER

Adds a new layer to the current technology.

Format

```
NEW LAYER layer_name
```

Arguments

```
layer_name (type STRING)
```

Specifies the name of layer to be created.

Named Arguments

```
[/GDSIInumber=] [/datatype=]
```

Specifies the GDSII number and datatype respectively.

```
[/red=] [/green=] [/blue=]
```

Numbers between 0 and 255 that specify the RGB value of the layer color.

Example

```
>new layer “LayerErr” /GDSIInumber=63 /datatype=0
```

Adds “LayerErr” with GDSII number 63 and datatype 0.

DELETE LAYER

Deletes the layer from technology. All objects on specified layer in all cells will be deleted. Operation cannot be undone.

Format

```
DELETE LAYER layer_name
```

Arguments

```
layer_name (type STRING)
```

Example

```
>delete layer “LayerErr”
```

Deletes “LayerErr” layer.

RENAME LAYER

Renames the layer.

Format

```
RENAME LAYER old_layer_name new_layer_name
```

Arguments

```
old_layer_name (type STRING)
```

Name of layer to be renamed.

```
new_layer_name (Type STRING)
```

New layer name.

Example

```
>rename layer “Met” “Metal1”
```

Layer “Met” is renamed to “Metal1”.

HIDE LAYER

Sets the specified layer or all layers except the active one to be invisible.

Format

```
HIDE LAYER [layer_name]
```

Arguments

```
[layer_name] (Type STRING)
```

Name of layer to be invisible. If name is not specified all layers except active one will be invisible.

Example 1

```
>hide layer "Met"
```

All objects on layer "Met" are hidden.

Example 2

```
>hide layer
```

Makes all layers except active one invisible.

SHOW LAYER

Makes the specified layer or all layers visible.

Format

```
SHOW LAYER [layer_name]
```

Arguments

```
[layer_name] (type STRING)
```

Name of layer to set visible. If the name is not specified, all layers will be visible.

Example 1

```
>show layer "Met"
```

Layer "Met" is visible.

Example 2

```
>show layer
```

Makes all layers visible.

SET_GDSII_NUMBER

Sets GDSII number for layer.

Format

```
SET_GDSII_NUMBER layer_name GDSII_number
```

Arguments

```
layer_name (type STRING)
```

Specifies name of layer.

```
GDSII_number (Type INTEGER)
```

Specifies new GDSII number for layer.

Example

```
>set_GDSII_number “Met1” 59
```

Sets GDSII number 59 for the layer “Met”.

SET_GDSII_DATATYPE

Sets GDSII datatype for the layer.

Format

```
SET_GDSII_DATATYPE layer_name GDSII_datatype
```

Arguments

```
layer_name (type STRING)
```

Specifies name of layer.

```
GDSII_datatype (Type INTEGER)
```

Specifies new GDSII data type for layer.

Example

```
>set_GDSII_datatype “Met1” 1
```

Sets GDSII datatype 1 for the layer “Met”.

STYLE WIRE

Sets wire style for the current layer similarly to **Setup→Technology→Layer/Wire→Wire Style** setting.

Format

```
STYLE WIRE [/width=value] [/angle=value]
[/extend | /flushend | /roundend]
[/extjoint | /filljoint | roundjoint | /miterjoint]
```

Named Arguments

```
[/width=value] (type INTEGER, FLOAT, DOUBLE)
```

Specifies default wire width for the active layer.

```
[/angle=value] (type INTEGER, FLOAT, DOUBLE)
```

Specifies default miter angle value for the active layer (valid only for /miterjoint style).

Switches

```
[/extend | /flushend | /roundend]
```

Specifies the default wire end style for the active layer.

```
[/extjoint | /filljoint | roundjoint | /miterjoint]
```

Specifies the default wire joint style for the active layer.

Example

```
>style wire /width = 0.5 /extend /roundjoint
```

LAYERPLAN LOAD

Loads layer plans from a specified disk file.

Note: Layer plan should be compatible with current technology.

Format

```
LAYERPLAN LOAD pathname
```

Arguments

```
pathname (Type STRING)
```

Specifies the name of a disk file that contains EXPERT layer plans.

Example

```
>layerplan load "d:\\temp\\asa.elp"
```

17.7.2: Layer Procedures

GET_EDITED_LAYER_NAME

Returns the name of the current layer.

Syntax

```
layer_name = GET_EDITED_LAYER_NAME()
```

Return Value

Name of the current layer (type `STRING`).

Example

```
>layer1 = get_edited_layer_name()
```

LAYER_EXISTS

Returns TRUE if the layer with specified name exists and FALSE if otherwise.

Syntax

```
ret = LAYER_EXISTS(layer_name)
```

Parameter

```
layer_name (type STRING)
```

Name of the layer.

Return Value

```
ret (type BOOLEAN)
```

TRUE if the layer with specified name exists, and FALSE otherwise.

Example

```
>display(layer_exists("Met1"))
```

GET_LAYER_LIST

Gets the sequence of layer names for the current project.

Syntax

```
layers = GET_LAYER_LIST()
```

Return Value

```
layers (type SEQUENCE)
```

Sequence of layer names.

Example

```
>layers = get_layer_list()
```

GET_LAYER_OBJECT_COUNT

Returns the number of objects in the specified cell and layer.

Syntax

```
obj_num = GET_LAYER_OBJECT_COUNT(cell_name, layer_name)
```

Parameters

```
cell_name (type STRING)
```

Full name of a cell.

```
layer_name (type STRING)
```

Name of the layer.

Return Value

```
obj_num (type INTEGER)
```

Number of objects in the specified cell and layer.

Example

```
>nobjs = get_layer_object_count(“mux4::inverter”, “Met1“)
```

GET_GDSII_NUMBER

Returns GDSII number of the specified layer.

Syntax

```
GDSII_num = GET_GDSII_NUMBER(layer_name)
```

Parameter

```
layer_name (type STRING)
```

Name of the layer.

Return Value

```
GDSII_num (type INTEGER)
```

GDSII number of the specified layer.

Example

```
>num = get_GDSII_number(“Met1“)
```

Gets the GDSII number of the layer “Met1”.

GET_GDSII_DATATYPE

Returns GDSII data type of the specified layer.

Syntax

```
GDSII_dt = GET_GDSII_DATATYPE(layer_name)
```

Parameter

```
layer_name (type STRING)
```

Name of the layer.

Return Value

```
GDSII_dt (type INTEGER)
```

GDSII data type of the specified layer.

Example

```
>GDSII_dt = get_GDSII_datatype("Met1")
```

Gets the GDSII datatype of the layer "Met1".

LAYER_SET_VIEW_ATTRS

Sets or clears the layer attribute.

Syntax

```
LAYER_SET_VIEW_ATTRS(layer_name, attr_type, flag)
```

Parameters

```
layer_name (type STRING)
```

Name of the layer. If "layer_name" is an empty string (""), then the specified attribute is set for all layers.

```
attr_type (predefined constants)
```

Can be one of:

```
LVA_VISIBLE – sets layer visibility
LVA_SELECTABLE – sets layer selectability
flag (type BOOLEAN)
```

TRUE to set the attribute or FALSE to clear the attribute.

Example

```
>layer_set_view_attrs("Met1", LVA_SELECTABLE, FALSE)
```

Makes the layer "Met1" unselectable.

17.8: Commands for Creating Objects

Note: Objects are created in the edited cell (and primitives are created in the active layer); see CELL OPEN and LAYER commands in Section 17.6: "Setup Commands".

17.8.1: Common Named Arguments and Switches

These named arguments and switches are used in all object creation commands.

Named Arguments

```
[/parent_cell=] type (STRING)
```

Specifies the name of the cell in which a search for the object with the specified attributes will be performed. Not valid if the switch /create from the named argument cluster [/create | /find | /delete] is specified. Note, when the named argument '/parent_cell' is used, objects attributes should be in the local coordinates of specified parent cell.

`[/property=({name1,value1, name2,value2,…,namen,valuen})] (Type SEQUENCE)`

- `{name1,value1, name2,value2,…,namen,valuen}` is a sequence of user-defined properties.
- `<namei,valuei>` is a property definition.

`[/text=({text1,text2,..., textn})] (Type SEQUENCE)`

The sequence of texts to be attached. Each text is represented by a sequence of text attributes

`texti = {x,y,string,height,width,slope,slant,align,mirror}`

See "TEXT" on page 17-45 for explanation of each attribute, and see "MODIFY TEXT" on page 17-73 for predefined text align style constants.

`[/replace=(layoutobject)] (type LAYOUTOBJECT)`

The layout object to be replaced by this object.

`[/net=net_name] (type STRING)`

Specifies the net to which the created object will be added.

Switches

`[/create | /find | /delete]`

- `/create` switch means that the new object will be added to cell.
- `/find` switch means that the object will be searched for.
- `/delete` switch means that the object will be deleted.

The default switch is determined by values of other named arguments and switches. The `/find` switch is used if switches `/select` or `/get` are present, otherwise the `/create` switch is used.

`[/select]`

- `/select` switch means that the object will be selected or deselected, according to object select mode defined by **select mode** command, or **Select→Selection Modes** setting.

`[/reselect /add /deselect]`

These switches temporary change the default select object mode.

- `/reselect` switch means that the object will be selected, and previously selected objects will be deselected.
- `/add` switch means that the object will be selected without a change in the selection state of other objects.
- `/deselect` switch means that the object will be deselected.

`[/get]`

- `/get` switch means that the new object (`/create`), or object that is searched for (`/find`) will be returned as a `LAYOUTOBJECT` type variable.

`[/pin]`

- `/pin` switch means that the object added by the named argument `/net` to the net will be used as a net pin in the instance connectivity (see Section 17.16: "Connectivity Commands"). The switch is ignored if the object does not belong to any net.

Note: Combinations of following switches and named arguments are illegal: `/delete` and `/get`, `/delete` and `/select`, `/delete` and `/replace`, `/find` and `/replace`.

17.8.2: Primitive Object Creation

Named arguments and switches applicable for each primitive object creation command described below.

Named Arguments

```
[/layer=layer_name] (type STRING)
```

Specifies the layer where the object will be created, deleted, or searched for. If the named argument `/layer` is not present, the object is created, deleted, or searched for in the active layer.

BOX

Creates, selects, or deletes a box.

Format

```
BOX xpos ypos xsize ysize
[layer=layername]
[/property=({name1,value1,name2,value2,…,namen,valuen})]
[/text=({text1,text2,...,textn})]
[/replace=layoutobject]
[net=netname]
[/create | /find | /delete]
[select]
[/reselect | /add | /deselect]
[/topolygon]
[/pin]
[/get]
```

Arguments

`xpos (type INTEGER, FLOAT, DOUBLE)`

- `X` coordinate of the left box corner.

`ypos (type INTEGER, FLOAT, DOUBLE)`

- `Y` coordinate of the left box corner.

`xsize (type INTEGER, FLOAT, DOUBLE)`

- Specifies `X` size of box.

`ysize (type INTEGER, FLOAT, DOUBLE)`

- Specifies `Y` size of box.

Named Arguments

```
[/layer=layer_name] (Type STRING)
```

See the description above on 17-40.

Switches

```
[/topolygon]
```

Specifies that object (created or searched for) will be converted into polygon.

See Section 17.8.1: "Common Named Arguments and Switches" for description of other named arguments and switches

Return Value

An object representing the layout object (type LAYOUTOBJECT) if the /get switch is used, otherwise NIL.

Example 1

```
>box 0 0 4 5
```

Creates the box in the active layer.

Example 2

```
>box 0 0 -4 -5
```

Creates the box in the active layer with attributes: xpos = -4, ypos = -5, xsize = 4, ysize = 5. If the size values are negative, then box attributes are recalculated as:

```
xpos = xpos - abs (xsize); xpos = xpos - abs (xsize); xsize = abs(xsize);
ysize = abs(ysize)
```

Example 3

```
>box 0 0 4 5 /layer = “CONT”
```

Creates the box in the layer CONT.

Example 4

```
>box 0 0 4 5 /create /select /reselect
>select mode /deselect
>box 0 0 4 5 /select
```

The first command creates the box in the active layer, selects it, and deselects the other objects. The second command sets the default select mode as deselect. The third command searches for the box, and deselects it according to default select mode.

Example 5

```
>box 0 0 4 5 /delete
```

Searches for the object with specified attributes in the active layer, and then deletes one.

Example 6

```
>box 0 0 4 5 /property=({“name1”, “10“, “name2”, “20“}) -
>/text=({{0,0,”text1”, 0.5, 0.5, 0, false, TA_BOTTOMLEFT, true}, -
>{0,5,”Text2”, 0.5, 0.5, 0, true, TA_TOPLEFT, false}})
```

Creates the object with specified properties and attached texts in the active layer.

Example 7

```
>old_box = (box 0 0 4 5 /create /get)
>box 0 0 5 6 /replace=(old_box)
```

The first command creates a box, and then returns one (type LAYOUTOBJECT). The second command replaces this object with the new one.

ELLIPSE

Creates, selects, or deletes a ellipse.

Format

```
ELLIPSE xcenter ycenter xradius yradius
[layer=layername]
[/property=({name1,value1,name2,value2,…,namen,valuen})]
```

```
[/text=({text1,text2,...,textn})]
[/replace=layoutobject]
[net=netname]
[/create | /find | /delete]
[select]
[/reselect | /add | /deselect]
[/topolygon]
[/pin]
[/get]
```

Arguments

`xcenter (type INTEGER, FLOAT, DOUBLE)`

- `X` coordinate of the ellipse center.

`ycenter (type INTEGER, FLOAT, DOUBLE)`

- `Y` coordinate of the ellipse center.

`xradius (type INTEGER, FLOAT, DOUBLE)`

- Specifies `X` radius of ellipse.

`yradius (type INTEGER, FLOAT, DOUBLE)`

- Specifies `Y` radius of ellipse.

Named Arguments

`[/layer=layer_name] (type STRING)`

See the description above on 17-40.

Switches

`[/topolygon]`

Specifies that the object (created or searched for) will be converted into a polygon.

See Section 17.8.1: "Common Named Arguments and Switches" for descriptions of other named arguments and switches.

Return Value

An object representing the layout object (type `LAYOUTOBJECT`) if the `/get` switch is used, otherwise NIL.

Example 1

```
>ellipse 0 0 4 5
```

Creates the ellipse in the active layer.

Example 2

```
>ellipse 0 0 4 5 /layer = "CONT"
```

Creates the ellipse in the layer `CONT`.

Example 3

```
>select all;
>ellipse 0 0 4 5 /create /select /reselect;
```

The first command selects all objects. The second command creates the ellipse in the active layer, selects it, and deselects the other objects.

Example 4

```
>ellipse 0 0 4 5 /delete
```

Searches for the object with specified attributes in the active layer and deletes it.

Example 5

```
>ellipse 0 0 4 5 /property=({“name1”, “10“, “name2”, “20“}) -
>/text=({{0,0,”text1”, 0.5, 0.5, 0, false, TA_BOTTOMLEFT, true}, -
>{0,5,”Text2”, 0.5, 0.5, 0, true, TA_TOPLEFT, false}})
```

Creates an object with specified properties and attached texts in the active layer.

Example 6

```
>old_ellipse = (ellipse 0 0 4 5 /create /get)
>ellipse 0 0 5 5 /replace=(old_ellipse)
```

The first command creates ellipse and returns one (type `LAYOUTOBJECT`). The second command replaces this object with the new one.

Example 7

```
>ellipse 0 0 4 5 /create /topolygon /select /add
```

Creates the ellipse, converts it to a polygon, and selects the last one. Selection status of other objects is not changed.

POLYGON

Creates, selects, or deletes a polygonal region.

Format

```
POLYGON {x1,y1,x2,y2,…,xn,yn}
[layer=layername]
[/property=({name1,value1,name2,value2,…,namen,valuen})]
[/text=({text1,text2,...,textn})]
[/replace=layoutobject]
[net=netname]
[/create | /find | /delete]
[select]
[/reselect | /add | /deselect]
[/topolygon]
[/pin]
[/get]
```

Arguments

$\{x_1,y_1,x_2,y_2,\ldots,x_n,y_n\}$ (type `SEQUENCE`)

Sequence of vertices of the polygon, where each pair of numeric values (x_i,y_i) is `X` and `Y` coordinates of `i-th` vertex accordingly. You may substitute any (x_i,y_i) pair by variable of `POINT` type (see examples below).

Named Arguments

`[/layer=layer_name] (type STRING)`

See the description above on 17-40.

See Section 17.8.1: "Common Named Arguments and Switches" for a description of other named arguments and switches

Return Value

An object representing layout object (type `LAYOUTOBJECT`) if `/get` switch is used, otherwise NIL.

Example 1

```
>v1_x = 0
>v1_y = 0
>polygon (v1_x),(v1_y), 4,0, 4,2, 2,2, 2,4, 0,4
```

or

```
>polygon {0,0, 4,0, 4,2, 2,2, 2,4, 0,4}
```

or

```
>pt = point_create(2, 2)
>polygon {0,0, 4,0, 4,2, (pt), 2,4, 0,4}
```

Creates the same polygon in the active layer.

Example 2

```
>polygon {0,0, 4,0, 4,2, 2,2, 2,4, 0,4} /layer = "POLY"
```

Creates a polygon in the layer `POLY`.

Example 3

```
>polygon {0,0, 4,0, 4,2, 2,2, 2,4, 0,4} /select
```

Searches for the object with specified parameters, and selects or deselects according to object select mode defined by **select mode** command or **Select→Selection Modes** setting.

Example 4

```
>polygon {0,0, 4,0, 4,2, 2,2, 2,4, 0,4} /create /select /add
```

Creates the object in the active layer and selects it (selection status of other objects is not changed).

Example 5

```
>polygon {0,0, 4,0, 4,2, 2,2, 2,4, 0,4} /delete
```

Deletes the object with specified attributes in the active layer, if it exists.

Example 6

```
>polygon {0,0,4,0,4,2,2,2,2,4,0,4} /property=({"name1", "nand1"})
```

Creates the object with specified property in the active layer.

Example 7

```
>polygon {0,0, 4,0, 4,2, 2,2, 2,4, 0,4} -
> /text=({{0,0,"text1", 0.5, 0.5, 0, false, TA_BOTTOMLEFT, true}, -
>         {0,2,"T2", 0.5, 0.5, 0, true, TA_TOPLEFT, false}})
```

Creates the object with specified attached texts in the active layer.

Example 8

```
>old_pol = (polygon {0,0, 4,0, 4,2, 2,2, 2,4, 0,4} /get)
>polygon {0,0,4,0,4,4,0,4} /replace=(old_pol)
```

The first command searches for an object with specified attributes in the active layer. The second command replaces this object with the new one.

TEXT

Creates, selects, or deletes a text.

Format

```
TEXT xpos ypos string
[/height=textheight]
[/width=textwidth]
[/slope=slope]
[layer=layername]
[/property=({name1,value1,name2,value2,…,namen,valuen})]
[/attachto=layoutobject]
[/detachfrom=layoutobject]
[/replace=layoutobject]
[/bottomleft | /bottomcenter | /bottomright |/middleleft | /middlecenter | /
middleright |
/topleft | /topcenter | /topright]
[/slant]
[/mirror]
[/polygonal]
[/proportional]
[/create | /find | /delete]
[select]
[/reselect | /add | /deselect]
[/get]
```

Arguments

`xpos (type INTEGER, FLOAT, DOUBLE)`

- `X` coordinate of the text position.

`ypos (type INTEGER, FLOAT, DOUBLE)`

- `Y` coordinate of the text position.

`string (type STRING)`

- Single line of text.

Named Arguments

`[/height=textheight] (type INTEGER, FLOAT, DOUBLE)`

Sets the height of a character. If the argument is not present, the default value from the registry will be used.

`[/width=textwidth] (Type INTEGER, FLOAT, DOUBLE)`

Sets the width of character. If the argument is not present, the default value from the registry will be used.

`[/slope=slope] (Type INTEGER, FLOAT, DOUBLE)`

Sets the slope of the text line in degrees. Default value for slope argument is 0.0 degree.

`[/layer=layer_name] (type STRING)`

See the description above on 17-40.

`[/attachto=layoutobject] (Type LAYOUTOBJECT)`

The layout object to which the text will be attached.

`[/detachfrom=layoutobject] (Type LAYOUTOBJECT)`

The layout object from which the text will be detached and added to the current edited cell.

Switches

```
[/bottomleft | /bottomcenter | /bottomright |
/middleleft | /middlecenter | /middleright |

/topleft | /topcenter | /topright]
```

Defines text alignment style with respect to its origin.

`[/slant]`

Indicates that text will be slanted (italic).

`[/mirror]`

Indicates that text will be mirrored with respect to the `X` axis.

`[/polygonal]`

It corresponds to the **Polygonal text** option of the **Numeric Input** panel for EXPERT'S E**dit→Create Object→Text** command. Combinations of switches `'/polygonal'` and `'get'`, `'/polygonal'` and `'/delete'`, `'/polygonal'` and `'attach'` are illegal for XI command text.

`[/proportional]`

Sets attributes proportional for polygonal texts. If the text is not polygonal (`/polygonal` is not present), then it does not have any meaning.

See Section 17.8.1: "Common Named Arguments and Switches" for descriptions of other named arguments and switches

Return Value

An object representing the layout object (type `LAYOUTOBJECT`) if the `/get` switch is used, otherwise NIL.

Example 1

```
>text 0 0 "Expert" /height = 0.5 /width = 0.5;
```

Creates the text "Expert" with specified attributes in the active layer. Default `/slope` is equal 0; default `/bottomleft` align style is used.

Example 2

```
>text 0 0 "Expert" /height = 0.5 /width = 0.5 /slant /mirror -
/layer = "INNOTE";
```

Creates the slanted and mirrored text in the layer "`INNOTE`".

Example 3

```
>text 0 0 "Expert" /height=0.5 /width=0.5 /create /select /reselect
```

Creates the text in the active layer, and selects it. Other objects are deselected.

Example 4

```
>text 0 0 "Expert" /height = 0.5 /width = 0.5 /delete;
```

Searches for the object with specified attributes in the active layer and deletes.

Example 5

```
>text 0 0 "Expert" /height = 0.5 /width = 0.5 -
>/property=({"name1", "text", "width", "0.5"})
```

Creates the object with specified properties in the active layer. Note that the named argument /text is not used for text object.

Example 6

```
>old_text=(text 0 0 "Expert" /height = 0.5 /width = 0.5 /create /get)
>text 0 0 "EXPERT_LISA" /height=0.5 /width = 0.5 /slant /slope = 90 -
>/replace=(old_text);
```

The first command creates text and returns one (type LAYOUTOBJECT). The second command replaces this object by the new text.

Example 7

```
>b1 = (box 0 0 5 5 /create /get);
>text 0 0 "EXPERT_LISA" /height = 0.5 /width = 0.5 /topleft -
/attachto = (b1);
```

The first command creates the box. The second command attaches the text "EXPERT_LISA" to this layout object (b1).

Example 8

```
>b2 = (box 0 0 5 5 -
/text=({{0,0,"text1", 0.5, 0.5, 0, false, TA_BOTTOMLEFT, true}}) -
/create /get)
>text 0 0 "text1" /height = 0.5 /width = 0.5 /topleft -
/detachfrom = (b2)
```

The first command creates the box with attached text. The second command detaches this text from this layout object (b2).

WIRE

Creates, selects, or deletes a wire. By default, wire attributes are defined by the corresponding layer. You can change wire attributes using named arguments and switches. The STYLE WIRE command is used for redefinition of the default wire attributes of the active layer.

Format

WIRE $\{x_1, y_1, x_2, y_2, \ldots, x_n, y_n\}$
[/width=width]
[/angle=angle]
[layer=layername]
[/property=($\{name_1, value_1, name_2, value_2, \ldots, name_n, value_n\}$)]
[/text=($\{text_1, text_2, ..., text_n\}$)]
[/replace=(layoutobject)]
[/extend | /flushend | /roundend]
[/extjoint | /filljoint | /roundjoint | /miter]

```
[/create | /find | /delete]
[select]
[/reselect | /add | /deselect]
[/topolygon]
[/get]
```

Arguments

$\{x_1, y_1, x_2, y_2, \ldots, x_n, y_n\}$ (type SEQUENCE)

Sequence of vertices of the wire, where each pair of numeric values (x_1,y_1) is X and Y coordinates of i-th vertex accordingly. You may substitute any (x_1,y_1) pair by a variable of POINT type (see examples below).

Named Arguments

```
[/width=width] (type INTEGER, FLOAT, DOUBLE)
```

Specifies the wire width distinction from default value for specified (/layer) or active layer.

```
[/angle=angle] (type INTEGER, FLOAT, DOUBLE)
```

Additional parameter of the wire joint style attribute. Sets the value of miter angle distinction from the default value for the specified (/layer) or active layer.

```
[/layer=layer_name] (type STRING)
```

See the description above on 17-40.

Switches

```
[/extend | /flushend | /roundend]
```

Specifies the wire end style distinction from the default value for specified (/layer) or active layer.

```
[/extjoint | /filljoint | /roundjoint | /miter]
```

Specifies the wire joint style distinction from the default value for specified (/layer) or active layer.

```
[/topolygon]
```

Specifies the object (created or searched for) that will be converted into a polygon.

See Section 17.8.1: "Common Named Arguments and Switches" for a description of other named arguments and switches

Return Value

An object representing the layout object (type LAYOUTOBJECT) if the /get switch is used, otherwise NIL.

Example 1

```
>wire {0,0, 10,0, 10,5, 5,5} /width = 1 /extend /extjoint;
```

Creates the wire with specified attributes in the active layer.

Example 2

```
>wire {0,0, 10,0, 10,5, 5,5} /width = 1 /flushend /filljoint -
>/layer="METAL1";
```

Creates the wire with specified attributes in the layer "METAL1".

Example 3

```
>select all;
```

```
>wire {0,0, 10,0, 10,5, 5,5} /width = 1 /flushend /filljoint -
>/create /select /add
```

The first command selects all objects. The second command creates the wire in the active layer and selects it. Selection status of other objects is not changed.

Example 4

```
>wire {0,0, 10,0, 10,5, 5,5} /width = 1 /flushend /filljoint /delete;
```

Searches for the object with specified attributes in the active layer and deletes one.

Example 5

```
>wire {0,0, 10,0, 10,5, 5,5} /width = 1 /extend /filljoint -
>/property=({"name1", "wire", "", "4,5"}) -
>/text=({{0,0,"text1", 0.5, 0.5, 0, false, TA_TOPLEFT, true}, -
>{0,5,"Text2", 0.5, 0.5, 0, true, TA_BOTTOMLEFT, false}});
```

Creates the object with specified properties and attached texts in the active layer.

Example 6

```
>old_wire = (wire {0,0, 10,0, 10,5, 5,5} /width=1 /extend /extjoint -
>/create /get);
polygon {0,0, 10,0, 10,5, 5,5} /replace=(old_wire);
```

The first command creates wire and returns one (type `LAYOUTOBJECT`). The second command replaces this object with the new polygon.

Example 7

```
>wire {0,0, 10,0, 10,5, 5,5} /flushend /miterjoint /create /topolygon;
```

Creates the wire and converts it to a polygon.

17.8.3: Instance Object Creation

INSTANCE

Creates, selects or deletes an instance of a cell.

Format

```
INSTANCE xpos ypos
[/cell=cellname]
[/scale=value]
[/rotate=value]
[/params=value]
[/parnames=value
[/property=({name1,value1,name2,value2,…,namen,valuen})]
[/text=({text1,text2,...,textn})]
[/replace=(layoutobject)]
[/net=netname]
[/name=instancename]
[/conn=connectivity]
[/horizontal | /vertical]
[/create | /find | /delete]
```

```
[select]
[/reselect | /add | /deselect]
[/pin]
[/get]
```

Arguments

`xpos (type INTEGER, FLOAT, DOUBLE)`

- `X` coordinate of the instance position.

`ypos (type INTEGER, FLOAT, DOUBLE)`

- `Y` coordinate of the instance position.

Named Arguments

`[/cell=cellname] (type STRING)`

Specifies the cell for instancing. If the named argument `/cell` is not present, then the instancing of the cell is assigned by **cell activate** command.

`[/scale=value] (type INTEGER, FLOAT, DOUBLE)`

Allows changing of the size of the cell instance. Default value is 1.

`[/rotate=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies the rotation angle. Defaut value is 0.

`[/params={param`$_1$`, param`$_2$`,...,param`$_n$`}] (type SEQUENCE)`

The sequence of parameter values for a parameterized cell. If the named parameter `/parnames` is not present, then the parameter's values should be in the exact order of parameters declared in the parameterized pcell master. Make sure that the types of parameters match. If the named argument `/params` is not present for a parametrized cell instance, then the cell instance is placed with default parameter values.

`[/parnames={name`$_1$`, name`$_2$`,...,name`$_n$`}] (type SEQUENCE)`

The sequence of parameter names of the parameterized cell. Pcell parameter's values defined by the named argument `/params` has to be in order of the parameter's names: parameter named `name`$_i$ has the value `param`$_i$. It is not necessary that the order and quantity of parameters defined by both `/params` and `/parnames` named arguments match those declared in the corresponding parameterized pcell master. If some parameters are missing the default parameter values from the pcell master are used.

`[/net=netname] (type STRING)`

Specifies the net to which this instance is added as single element. The instance cell should not have any external nets. Use named argument `/conn` or connectivity creation commands (see Section 17.16: "Connectivity Commands") for adding instance's connectivity on nets.

`[/name=instancename] (type STRING)`

Assigns the name for the instance object. The name should be unique to the cell where this instance is laid out.

`[/conn=connectivity] (type SEQUENCE)`

Specifies how this instance connected to the current cell nets. The information is in the form:

```
{instance_cell_net_name->current_cell_net_name},
```

where each element of the sequence above consist of two parts linked by set of characters '->': 'instance_cell_net_name' is the external net name of the instanced cell and current_cell_net_name' is the net name of the current cell correspondingly.

Switches

```
[/horizontal | /vertical]
```

`/horizontal` specifies mirror over the y axis; `/vertical` specifies mirror over the x axis.

See Section 17.8.1: "Common Named Arguments and Switches" for a description of other named arguments and switches

Return Value

An object representing the layout object (type `LAYOUTOBJECT`) if the `/get` switch is used, otherwise NIL.

Example 1

```
>instance 0 0 /rotate = 90 /vertical;
```

Creates the instance of current activated cell.

Example 2

```
>instance 0 0 /rotate = 45 /cell = "mux1"
```

Creates the instance of the cell "`mux1`" with nonorthogonal rotation angle.

Example 3

```
>select all
>select mode /reselect;
>instance 0 0 /rotate = 90 /cell = "nand2" /create /select;
```

Creates the instance of the cell `nand2` and selects it. Other objects are deselected according to default selection mode `reselect`.

Example 4

```
>instance 0 0 /horizontal /cell = "inverter" /delete;
```

Searches for the object with specified attributes and deletes it.

Example 5

```
>instance 0 0 /cell = "mux" -
>/property=({"name1", "text", "cell", "nand2"}) -
>text=({{0,0,"text1", 0.5, 0.5, 0, false, TA_TOPLEFT, true},-
>{0,5,"Text2", 0.5, 0.5, 0, true, TA_BOTTOMLEFT, false}})
```

Creates the instance with specified properties and attached texts.

Example 6

```
>old_instance = (instance 0 0 /cell = "nand" /create /get)
>instance 0 0 /cell = "nand2" /replace=(old_instance)
```

The first command creates the instance of the cell `nand` and returns one (type `LAYOUTOBJECT`). The second command replaces this object by the instance of cell `nand2`.

ARRAY

Creates, selects or deletes a cell instance array.

Format

```
ARRAY xpos ypos
[/cell=cellname]
```

```
[/scale=value]
[/rotate=value]
[/params=value]
[/parnames=value]
[/row=value]
[/col=value]
[/xdeltarow=value]
[/ydeltarow=value]
[/xdeltacol=value]
[/ydeltacol=value]
[/property=({name1,value1,name2,value2,…,namen,valuen})]
[/text=({text1,text2,...,textn})]
[/replace=(layoutobject)]
[/net=netname]
[/name=instancename]
[/conn=connectivity]
[/horizontal | /vertical]
[/create | /find | /delete]
[select]
[/reselect | /add | /deselect]
[/pin]
[/get]
```

Named Arguments

`[/row=value] (type INTEGER)`

Specifies the number of rows in a cell instance array. The value must be greater or equal 1. Default value is 1.

`[/row=value] (type INTEGER)`

Specifies the number of columns in a cell instance array. The value must be greater or equal 1. Default value is 1.

`[/xdeltarow=value] (type INTEGER, FLOAT, DOUBLE)`

Sets `X` shift for array rows. Default value is 0 (orthogonal array).

`[/ydeltarow=value] (type INTEGER, FLOAT, DOUBLE)`

Sets `Y` shift for array rows. Default value is the Y size of the cell instance (Y size of a cell bounding box after instance transformation was applied. The latter is specified by `/scale`, `/rotate`, and `/horizontal` or `/vertical` named arguments and switches).

`[/xdeltacol=value] (type INTEGER, FLOAT, DOUBLE)`

Sets `X` shift for array columns. Default value is the X size of the cell instance (X size of cell bounding box after instance transformation was applied; the latter is specified by `/scale`, `/rotate`, and `/horizontal` or `/vertical` named arguments and switches).

`[/ydeltacol=value] (type INTEGER, FLOAT, DOUBLE)`

Sets `Y` shift for array columns. Default value is 0 (orthogonal array).

See Section 17.8.1: "Common Named Arguments and Switches" and `INSTANCE` command for description of the other named arguments and switches

Note: If the number of rows or columns is not equal to 1, then only orthogonal rotation angles are allowed.

Return Value

An object represents the layout object (type `LAYOUTOBJECT`) if the `/get` switch is used, otherwise NIL.

Example 1

```
>cell activate "inverter";
>array 0 0 /row = 2 /col = 3 /ydeltarow = 20 /xdeltacol = 12;
```

Creates the array 2x3 of activated cell `inverter`.

Example 2

```
>array 40 0 /cell = “nand2” /row = 3 /col = 4 -
/ydeltarow = 12 /xdeltacol = 20 /rotate = 90;
```

Creates the array 3x4 of the cell `nand2` with rotation angle 90.

17.8.4: Device Instance Object Creation

CONTACT

Creates, modifies and deletes the instance of a predefined contact template.

Format

```
CONTACT contact_name xpos ypos
[/width=value] [/length=value]
[/topleft | /topcenter | /topright | /middleleft | /middlecenter |
 /middleright |
/bottomleft | /bottomcenter | /bottomright | /justification=value]
[/rows= value] [/cols=value]
[/deltarow=value] [/deltacol=value]
[/mirror=value] [/rotate=value]
[/property=({name1,value1,name2,value2,…,namen,valuen})]
[/text=({text1,text2,...,textn})]
[/parent_cell=value]
   [create | /find | /delete]
[/replace=(layoutobject)]
[/select] [/reselect | /add | /deselect]
   [/get]
```

Arguments

- `contact_name (type STRING)`: Name of the contact template stored in the technology.
- `xpos (type INTEGER, FLOAT, DOUBLE)`: X coordinate of the contact position.
- `ypos (type INTEGER, FLOAT, DOUBLE)`: Y coordinate of the contact position.

Named Arguments

- `[/width = value] (type INTEGER, FLOAT, DOUBLE)`: Specifies the contact width. If not present, the wire width is defined by the contact template.

- [/length = value] (type INTEGER, FLOAT, DOUBLE): Specifies the contact length. If not present, the contact length is defined by a contact template.

[/topleft | /topcenter | /topright | /middleleft | /middlecenter |
/middleright |
/bottomleft | /bottomcenter | /bottomright | /justification=value]

The cluster of named arguments modifies the default contacts array justification defined in the contact template. The following predefined constants (switches) can be used: CJT_TOPLEFT (/topleft), CJT_TOPCENTER (/topcenter), CJT_TOPRIGHT (/topright), CJT_MIDDLELEFT (/middleleft), CJT_MIDDLECENTER (/middlecenter), CJT_MIDDLERIGHT (/middleright), CJT_BOTTOMLEFT (/bottomleft), CJT_BOTTOMCENTER (/bottomcenter), CJT_BOTTOMRIGHT (/bottomright).

- [/rows = value] (type INTEGER): Specifies the number of rows in the contact array. If not present, the number of rows is defined by contact.
- [/cols = value] (type INTEGER): Specifies the number of columns in the contact array. If not present, the number of columns is defined by contact template.
- [/deltarow=value] (type INTEGER, FLOAT, DOUBLE): Specifies the distance between rows of contacts in the contact array. If not present, the default value from contact template is used.
- [/deltacol=value] (type INTEGER, FLOAT, DOUBLE): Specifies the distance between columns of contacts in the contact array. If not present, the default value from contact template is used.

See Section 17.8.1: "Common Named Arguments and Switches" for a description of other named arguments and switches

Return Value

An object representing the layout object (type LAYOUTOBJECT) if the /get switch is used, otherwise NIL.

Example

```
>contact "M1_P" -2 5 /width=1 /length=1;
```

Creates the new instance of contact "M1_P" in the position (-2;5) with width=1 and length=1.

17.8.5: Floorplan Object Creation

FLOORPLAN CHIP

Sets chip attributes.

Format

```
FLOORPLAN CHIP
[name=value]
[/posx=value]
[/posy=value]
[/sizex=value]
[/sizey=value]
[/fixsizex | /nofixsizex]
[/fixsizey | /nofixsizey]
[/ratiox=value]
[/ratioy=value]
[/fixratio | /nofixratio]
```

```
[/minarea=value]
[/maxarea=value]
[/fixarea | /nofixarea]
[/blockbordespacing=value]
[/blockspacing=value]
[/pinsizex=value]
[/pinsizey=value]
[/padsizex=value]
[/padsizey=value]
```

Named Arguments

`[/name=value] (type STRING)`

Specifies the name of the chip.

`[/posx=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies the X coordinate of the chip position.

`[/posy=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies the Y coordinate of the chip position.

`[/sizex=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies the horizontal chip size.

`[/sizey=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies the vertical chip size.

`[/ratiox=value] [/ratioy=value] (type INTEGER)`

Specifies the aspect ratio of horizontal and vertical chip size.

`[/minarea=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies minimal value for chip area.

`[/maxarea=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies maximal value for chip area.

`[/blockborderspacing=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies offset of block shape from chip boundary.

`[/blockspacing=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies space value between blocks.

`[/pinsizex=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies X pin size.

`[/pinsizey=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies Y pin size.

`[/padsizex=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies X pad size.

`[/padsizey=value] (type INTEGER, FLOAT, DOUBLE)`

Specifies Y pad size.

Switches

`[/fixsizex | /nofixsizex]`

- `/fixsizex` prevents `sizex` from modifying.
- `/nofixsizex` allows to modify `sizex`.

`[/fixsizey | /nofixsizey]`

- `/fixsizey` prevents `sizey` from modifying.
- `/nofixsizey` allows to modify `sizey`.

`[/fixratio | /nofixratio]`

- `/fixratio` prevents `ratiox` and `ratioy` from modifying.
- `/nofixratio` allows to modify `ratiox` and `ratioy`.

`[/fixarea | /nofixarea]`

- `/fixarea` prevents to change chip area.
- `/nofixarea` allows to change chip area.

Return Value

None

FLOORPLAN BLOCK

Creates, selects, or deletes a floorplan block.

Note: This command is obsolete. Any library cell can be used in place of old floorplan block. Use the command `'INSTANCE'` (see Section 17.8.3: "Instance Object Creation") to place a block into an active cell view and commands for connectivity creation and modification (see Section 17.16: "Connectivity Commands").

FLOORPLAN PAD

Creates, selects, or deletes a floorplan pad.

Note: This command is obsolete. Use primitive layout objects with the named argument `'/net'` and switch to `'/pin'`. See Section 17.8.2: "Primitive Object Creation".

1FLOORPLAN PIN

Creates, selects, or deletes a pin in the floorplan block. Pins are used to connect block terminals.

Note: This command is obsolete. Use primitive layout objects with the named argument `'/net'` and switch to `'/pin'`. See Section 17.8.2: "Primitive Object Creation".

17.8.6: Complex Object Creation

MULTIPATH

Creates a multiwire (complex layout object containing wires on different layers connected by appropriate contact cells).

Note: Technology for the current project must contain "contact" statements for layers to be contacted.

Format

MULTIPATH ({$wire_1$,$wire_2$,…,$wire_n$})

Arguments

{$wire_1$,$wire_2$,…,$wire_n$} (type SEQUENCE)

Sequence of wires to be contacted by appropriate contact cells. Each element can be either a layout object wire (type LAYOUTOBJECT), or sequence of wire attributes. The latter form is represented as:

$wire_i$={{vertices},layer,width,endstyle,joinstyle}

where {vertices} is a sequence of wire vertices (see "WIRE" on page 17-47), layer is the name of the layer, width is the wire width, endstyle is the wire end style, and joinstyle is the wire joint style (see pre-defined constants in "MODIFY WIRE" on page 17-74).

Note: The last vertex of each wire must coincide with the first vertex of next wire.

Return Value

None.

Example 1

```
>pathstruct = {{{0,0, 10,0}, "METAL1", 1.0, FLUSH_END, EXT_JOINT}, -
>{{10,0, 10,10}, "METAL2", 2.0, FLUSH_END, EXT_JOINT}, -
>{{10,10, 0,10}, "METAL1", 3.0, FLUSH_END, EXT_JOINT}};
>multipath (pathstruct);
```

Creates a multiwire from wires running in the layers METAL1 and METAL2, and appropriate contact instances.

Example 2

```
>pathstruct = {{{0,0, 10,0}, "METAL1", 1.0, FLUSH_END, EXT_JOINT}, -
>{{10,10}, "METAL2", 2.0, FLUSH_END, EXT_JOINT}, -
>{{0,10}, "METAL1", 3.0, FLUSH_END, EXT_JOINT}};
>multipath (pathstruct);
```

Creates the same multiwire as in Example 1. Note the last vertices of wires do not coincide with the first vertices of next wires. In this case, the last vertex of the wire is added to the next wire as the first vertex.

Example 3

```
>w1 = (wire {0, 0, 0, 20} /width = 1 /flushend /extjoint -
> /layer="METAL1" /create /get);
```

```
>w2 = (wire {0, 20, 20, 20} /width = 4 /flushend /extjoint -
> /layer="METAL2" /create /get);
>multipath ({w1, w2});
```

Creates the same multiwire as in Example 1. Each wire is presented by layout object wire.

Example 4

```
>multipath ({(wire {0, 0, 0, 20} /width = 1 /flushend /extjoint -
> /layer="METAL1" /create /get), -
> (wire {0, 20, 20, 20} /width = 4 /flushend /extjoint -
> /layer="METAL2" /create /get)});
```

Creates the same multiwire as in Example 1. Each wire is presented by layout object wire.

GROUP CREATE

Creates group from selected objects

Format

```
GROUP CREATE
```

GROUP DESTROY

Deletes selected object's group (group members are not deleted from layout).

Format

```
GROUP DESTROY [/all]
```

Switches

```
[/all]
```

If specified, all groups in the current cell will be destroyed; selected group will be destroyed otherwise.

PWIRE

Creates, modifies, deletes or destroys a complex Pwire object.

Format

```
PWIRE {vertices} [/name = pwire_name]
[/template = pwire_template_name]
[/layer = layername] [/width = value]
[/extend | /flushend | /roundend]
[/extjoint | /filljoint | /roundjoint | /miterjoint]
[/offset = value] [/offsetcenter | /offsetleft | /offsetright]
[/enclwires = {enclwires}]
[/offsetwires = {offsetwires}]
[/boxes = {boxes}]
[/create | /modify | /delete | /split]
```

Arguments

```
{vertices} (type SEQUENCE)
```

Sequence of vertices of the master wire object.

Named Arguments

- [/name = pwire_name] (type STRING): Specifies a unique name for the Pwire object. If not presented, the system defines a unique name and assigns it to the new object.
- [/template = pwire_template_name](type STRING): Specifies the Pwire template stored in the technology as a basis for the Pwire object. The Pwire template contains all necessary attributes for the Pwire object except vertices and name.
- [/layer = layername] (type STRING): Specifies a layer for the master wire object. If not presented, a layer from the Pwire template is used if the last one presented; current active layer is used otherwise.
- [/width = value] (type INTEGER, FLOAT, DOUBLE): Specifies the the master wire width. If not presented, the wire width is defined by the Pwire template; the default value for the specified (/layer) or active layer is in the specified order.
- [/extend | /flushend | /roundend]: Specifies the wire end style. If not presented, the wire end style is defined by the Pwire template; default value for the specified (/layer) or active layer is in the specified order.
- [/extjoint | /filljoint | /roundjoint | /miterjoint]: Specifies the wire joint style. If not presented, the joint end style is defined by the Pwire template; default value for the specified (/layer) or active layer is in the specified order.
- [/offset = value] (type INTEGER, FLOAT, DOUBLE): Specifies the offset of the created master wire from input vertices defined by the {vertices} argument. Default offset 0.
- [/offsetcenter | /offsetleft | /offsetright]: Specifies how to apply offset to the master Pwire.
  - /offsetcenter (default) means offset between centerline of the created path and input vertices.
  - /offsetleft means offset between left edge of the created path and input vertices.
  - /offsetright means offset between right edge of the created path and input vertices.
- [/enclwires = {enclwires}] (type SEQUENCE): Specifies Pwire subparts of the enclosure wire type if different from the enclosure wires defined in the Pwire template. See Section “PWIRE_TEMPLATE” for a description of the format for enclosure wire subparts.
- [/offsetwires = {offsetwires}] (type SEQUENCE): Specifies Pwire subparts of the offset wire type if different from the offset wires defined in the Pwire template. See cSection “PWIRE_TEMPLATE” for a description of the format for offset wire subparts.
- [/boxes = {boxes}] (type SEQUENCE): Specifies Pwire subparts of box type if different from boxes defined in the Pwire template. See Section “PWIRE_TEMPLATE” for a description of the format for box subparts.

  [/create | /modify | /delete | /split]
  - /create (default) means the new Pwire object will be created.
  - /modify means the Pwire object will be modified. The named argument /name should be present.
  - /delete means the Pwire object will be deleted. The named argument /name should be present.
  - /split means the Pwire object will be split into regular wire and box objects. The named argument /name should be present.

Example

```
>pwire {-10,-10, -10,20, 20,20, 20,-10, -8,-10} /template =   "GuardRing_2";
```

Creates a new Pwire object in the current cell, supposing the Pwire template "GuardRing_2" exists in the technology.

PWIRE_TEMPLATE

Adds, modifies, deletes the Pwire template in the technology.

Format

```
PWIRE_TEMPLATE pwire_template_name
/layer = layername [/width = value]
[/extend | /flushend | /roundend]
[/extjoint | /filljoint | /roundjoint | /miterjoint]
[/centeroffset | /leftoffset | /rightoffset]
[/enclwires = {enclwires}]
[/offwires = {offwires}]
[/boxes = {boxes}]
```

Arguments

- `pwire_template_name (type STRING)`: Specifies Pwire template name.
- `/layer = layername (type STRING)`: Specifies layer for master wire object
- `[/width = value] (type INTEGER, FLOAT, DOUBLE)`: Specifies the master wire width. If not presented, the default wire width for specified layer is used.
- `[/extend | /flushend | /roundend]`: Specifies the wire end style. If not presented, the default wire end style for specified layer is used.
- [/extjoint | /filljoint | /roundjoint | /miterjoint]: Specifies the wire joint style. If not presented, the default joint end style for specified layer is used.
- `[/enclwires = {enclwires}] (type SEQUENCE)`: Specifies Pwire subparts of enclosure wire type.
- `[/offsetwires = {offsetwires}] (type SEQUENCE)`: Specifies Pwire subparts of offset wire type.
- `[/boxes = {boxes}] (type SEQUENCE)`: Specifies Pwire subparts of box type.

There are set of predefined Xi attributes used for definition of Pwire subparts, such as:

- `PWA_BEGIN_OFFSET`: Defines offset for subpart objects with respect to the first vertex of the master wire object (value type `INTEGER`, `FLOAT`, `DOUBLE`). If a value is positive, the subpart starts before first edge of master wire; if negative - after.
- `PWA_BEGIN_CORNER_OFFSET`: Defines the shift for the subpart boxes at the start of each internal edge (value type `INTEGER`, `FLOAT`, `DOUBLE`). By default (corner offset 0) the boxes are placed in the internal vertices.
- `PWA_DISTRIBUTION`: Defines the distribution of boxes (predefined constants: `BDT_EVEN`, `BDT_MIN`).
- `PWA_ENCLOSURE`: Defines width of the enclosure wires.
- `PWA_END_OFFSET`: Defines the offset for subpart objects with respect to the last vertex of the master wire object (value type `INTEGER`, `FLOAT`, `DOUBLE`). If the value is positive, the subpart ends after the last edge of the master wire; if negative - before.
- `PWA_END_CORNER_OFFSET`: Defines the shift for the subpart boxes at the end of each internal edge (value type `INTEGER`, `FLOAT`, `DOUBLE`).
- `PWA_JUSTIFICATION`: Controls the offset of subparts (predefined constants: `PWJ_LEFT`, `PWJ_CENTER`, `PWJ_RIGHT`).
- `PWA_LAYER`: Defines the subpart layer (value type `STRING`).

- PWA_LENGTH: Defines the length of the box in the subpart of type box (value type INTEGER, FLOAT, DOUBLE).
- PWA_OFFSET: Defines the separation from the master wire (value type INTEGER, FLOAT, DOUBLE). Valid for offset wires and boxes.
- PWA_SPACE: Defines the minimum space between boxes for the box subparts (value type INTEGER, FLOAT, DOUBLE).
- PWA_WIDTH: Defines the width of the wire in the subparts of type offset wire, or the width of the box in the subpart of type box (value type INTEGER, FLOAT, DOUBLE).

Each subpart is described as a sequence of attributes with values of subparts parameters.

	Enclosure Wire	Offset Wire	Box
Attributes	PWA_LAYER, PWA_BEGIN_OFFSET, PWA_END_OFFSET, PWA_ENCLOSURE	PWA_LAYER, PWA_BEGIN_OFFSET,PWA_END_OFFSET, PWA_WIDTH, PWA_OFFSET, PWA_JUSTIFICATION	PWA_LAYER, PWA_BEGIN_OFFSET, PWA_END_OFFSET, PWA_WIDTH, PWA_LENGTH, PWA_OFFSET, PWA_JUSTIFICATION, PWA_BEGIN_CORNER_OFFSET, PWA_END_CORNER_OFFSET, PWA_SPACE, PWA_DISTRIBUTION

Example

```
pwire_template " GuardRing_2"  /layer = "contact" /width = 3 /extend /
extjoint
/boxes = ({
{attr_create(PWA_LAYER,       "METAL1"),
attr_create(PWA_WIDTH,        1),
attr_create(PWA_LENGTH,       1),
attr_create(PWA_BEGIN_OFFSET, 0),
attr_create(PWA_END_OFFSET,   0),
attr_create(PWA_OFFSET,       0),
attr_create(PWA_JUSTIFICATION,PWJ_CENTER),
attr_create(PWA_SPACE,        1),
attr_create(PWA_DISTRIBUTION, BDT_EVEN),
attr_create(PWA_BEGIN_CORNER_OFFSET, 0),
attr_create(PWA_END_CORNER_OFFSET,   0)}});
```

Creates the Pwire template GuardRing_2 with the box subpart.

17.9: Select Commands

Additional ways of selection are given in Section 17.8: “Commands for Creating Objects”.

DESELECT ALL

Deselects all objects in the edited cell.

Format

```
DESELECT ALL
```

DESELECT OBJECT

Deselects an object, or a sequence of objects.

Format

```
DESELECT OBJECT lobject
```

Arguments

```
lobject (type LAYOUTOBJECT, SEQUENCE)
```

Layout object or a sequence of layout objects to be selected. Use commands for creating objects (Section 17.8: "Commands for Creating Objects") with the switch `/get` or `SEARCH` procedure (Section 17.11: "Search Commands and Procedures") to obtain layout objects.

Example

```
>pols = search(SEARCH_POLYGON, {}, WHOLE_CELL, FALSE, FALSE, TRUE)
>deselect object (bb)
```

Deselects all polygons in the current cell.

INVERT SELECTION

Toggles the selection state for all objects in the edited cell.

Format

```
INVERT SELECTION
```

SELECT ALL

Selects all objects in the edited cell.

Format

```
SELECT ALL
```

SELECT BOX

Selects layouts objects depending on the `box mode`.

Format

```
SELECT BOX xpos ypos xsize ysize
[/select | /reselect | /deselect]
```

Arguments

```
xpos (type INTEGER, FLOAT, DOUBLE)
ypos (type INTEGER, FLOAT, DOUBLE)
xsize (type INTEGER, FLOAT, DOUBLE)
ysize (type INTEGER, FLOAT, DOUBLE)
```

`(xpos, xpos)` is the lower left corner of selection box and (x_{size}, y_{size}) is size of box.

Switches

```
[/select | /reselect | /deselect]
```

These switches temporary change the default select object mode.

- `/select` switch means that the object will be selected without a change in the selection state of other objects.
- `/reselect` switch means that the object will be selected, and previously selected objects will be deselected.
- `/deselect` switch means that the object will be deselected.

SELECT INSTANCES

Selects instances of cells with the specified names.

Format

```
SELECT INSTANCES [{cell_name}]
```

Arguments

```
[{cell_name}] (type SEQUENCE)
```

{cell_name} = {$cell\_name_1$, …, $cell\_name_n$} is a sequence of cells which instances will be selected. If the argument is not specified, then all cell instances in the edited cell will be selected.

Example

```
>select instances
```

All instances in the edited cell are selected.

SELECT LAYERS

Selects all objects in the layers.

Format

```
SELECT LAYERS [{layer_name}]
```

Arguments

```
[{layer_name}] (type SEQUENCE)
```

{layer_name} = {$layer\_name_1$, …, $layer\_name_n$} is a sequence of layer which objects will be selected. If argument is not specified then all objects on the active layer will be selected.

Example

```
>select layers {“Metal”}
```

All objects on the layer “Metal” are selected.

SELECT NEWEST

Selects the new object in the edited cell produced by the `create` or `modify` commands.

Format

```
SELECT NEWEST
```

SELECT OBJECT

Selects an object, or a sequence of objects.

Format

```
SELECT OBJECT lobject
```

Arguments

```
lobject (type LAYOUTOBJECT, SEQUENCE)
```

Layout object or a sequence of layout objects to be selected. Use commands for creating objects (Section 17.8: "Commands for Creating Objects") with the switch `/get` or `FIND OBJECTS` procedure (Section 17.11: "Search Commands and Procedures") to obtain layout objects.

Example

```
>bb = (box 0 0 5 5 /create /get)
>select object (bb)
```

First command creates box on the active layer with specified attributes and returns it as variable "`bb`". Second command selects specified box.

SELECT TYPE

Selects objects of the specified type in the current cell.

Format

```
SELECT TYPE object_type
```

Arguments

```
object_type (predefined constants)
```

Specifies the type of layout objects for selecting. The following predefined constants are applicable

`LOT_ANY_OBJECT`

Selects all objects, equivalently `'SELECT ALL'` commands.

`LOT_ANY_SHAPE`

Selects all, but instance and arrays.

`LOT_BOX`

Selects only boxes.

`LOT_ELLIPSE`

Selects only ellipces.

`LOT_REGION`

Selects regions.

`LOT_WIRE`

Selects wires.

`LOT_TEXT`

Selects texts.

`LOT_INSTANCE`

Selects instances and arrays.

`LOT_ARRAY`

Selects arrays.

Example

```
>select object (LOT_BOX)
```

Selects all boxes in the current cell.

17.10: Edit Commands

COPY LAYER

Copies the set of selected objects into the layer.

Format

```
COPY LAYER [layer_name]
```

Arguments

```
[layer_name] (type STRING)
```

The name of a layer (optional parameter; if it is absent, objects are copied into the active layer in the edited cell).

Example

```
>copy layer "Metal21"
```

Selected objects are copied to layer `"Metal1"`.

COPY SELECTION

Copies the selected objects from the edited cell onto EXPERT'S clipboard.

Format

```
COPY SELECTION
```

CUT SELECTION

Moves the selected objects from the edited cell onto EXPERT'S clipboard.

Format

```
CUT SELECTION
```

DELETE SELECTION

Deletes the selected objects of the edited cell.

Format

```
DELETE SELECTION
```

DUPLICATE SELECTION

Duplicates the set of selected objects of the edited cell.

Format

```
DUPLICATE SELECTION shift_x yshift_y [repeat_x] [repeat_y]
```

Arguments

```
shift_x (type INTEGER, FLOAT, DOUBLE)
```

Displacements in `X` direction.

```
shift_y (type INTEGER, FLOAT, DOUBLE)
```

Displacements in `Y` direction.

```
repeat_x (type INTEGER)
```

Number of replication steps in `X` direction.

```
repeat_y (type INTEGER)
```

Number of replication steps in Y direction.

Example 1

```
>duplicate selection 0.5 1.0
```

Creates a copy of the selected set displaced by (x=0.5, y=1.0) from the original set.

Example 2

```
>duplicate selection 0.5 1.0 2 4
```

Replicates the selected set in array fashion, making 2 columns and 4 rows, with a spacing of 0.5 between columns, and 1.0 between rows.

Example 3

```
>duplicate selection 0.5 1.0 1 1
```

Does nothing.

MIRROR SELECTION

Mirrors the set of selected objects vertically or horizontally.

Format

```
MIRROR SELECTION [/x=value] [/y=value] [/duplicate]
```

Named Arguments

```
[/x=] (type INTEGER, FLOAT, DOUBLE)
```

Specifies vertical axis for the reflection of the selected objects with respect to it.

```
[/y=] (type INTEGER, FLOAT, DOUBLE)
```

Specifies horizontal axis for the reflection of the selected objects with respect to it.

Note: Using both named arguments in the command `MIRROR SELECTION` is illegal.

Switches

```
[/duplicate]
```

If specified then new objects will be created from selected ones and mirrored according to specified axis.

Example 1

```
>mirror selection /x=4
```

Mirror with respect to vertical axis at x=4.

Example 2

```
>mirror selection /y=7
```

Mirror with respect to horizontal axis at y=7.

MOVE LAYER

Moves the set of selected objects into the layer.

Format

```
MOVE LAYER [layer_name]
```

Arguments

```
[layer_name] (type STRING)
```

The name of a layer (optional parameter; if it is absent, objects are moved into the active layer in the edited cell).

Example

```
>move layer “Metal2”
```

Selected objects are moved to layer “Metal2”.

MOVE SELECTION

Moves the set of selected objects of the edited cell.

Format

```
MOVE SELECTION shift_x shift_y
```

Arguments

```
shift_x (type INTEGER, FLOAT, DOUBLE)
```

Displacements in X direction.

```
shift_y (type INTEGER, FLOAT, DOUBLE)
```

Displacements in Y direction.

Example

```
>move selection 0.5 1
```

Moves the selected set by (x=0.5, y=1) from the original position.

ORIGIN

Sets a new origin for the current cell.

Format

```
ORIGIN x y
```

Arguments

```
x (type INTEGER, FLOAT, DOUBLE)
y (type INTEGER, FLOAT, DOUBLE)
```

(x, y) are the coordinates for the new cell origin.

Example

```
>origin 20 30
```

PASTE LAYER

Inserts the set of objects from EXPERT'S clipboard into the layer

Format

```
PASTE LAYER [layer_name]
```

Arguments

```
[layer_name]
```

The name of a layer (optional parameter; if it is absent, objects are pasted into the active layer).

Example

```
>paste layer "Metal1"
```

PASTE SELECTION

Inserts the set of objects from the EXPERT'S clipboard into the edited cell.

Format

```
PASTE SELECTION x y
```

Arguments

```
x (type INTEGER, FLOAT, DOUBLE)
y (type INTEGER, FLOAT, DOUBLE)
```

(x, y) is a position, where the objects will be inserted.

Example

```
>paste selection 10, 5
```

ROTATE SELECTION

Rotates the set of selected objects of the edited cell.

Format

```
ROTATE SELECTION x y [angle] [/duplicate]
```

Arguments

```
x  (type INTEGER, FLOAT, DOUBLE)
y  (type INTEGER, FLOAT, DOUBLE)
```

(x, y) is the center of rotation.

```
[angle]  (type INTEGER, FLOAT, DOUBLE)
```

The angle of rotation. (Optional; 90 is default)

Switches

```
[/duplicate]
```

If specified, then new objects will be created as a copy of selected ones and rotated according to specified parameters.

Example

```
>rotate selection 0 0 180
```

REDO

Performs one step of redo.

Format

```
REDO
```

STRETCH DESELECT_ALL

Deleselects segments and objects selected for stretch operation.

Format

```
STRETCH DESELECT_ALL
```

Arguments

None.

STRETCH SELECT

Selects or deselects object segments or whole object for stretch operation.

Format

```
STRETCH SELECT [lobject] [/box=r] [/point=pt]
[/segment={pt1, pt2}] [/select | /reselect | /deselect]
```

Arguments

- `lobject (type LAYOUTOBJECT)`: Specifies the layout object part of which will be selected for stretch. The argument is ignored if selected by box mode (the `/box` argument presented).

Named Arguments

Selection or deselection of stretch parts can be performed by one of the following methods: by box (`/box` argument), by point (`/point` argument is presented and `lobject` specified), and by segment (`/segment` argument is presented and `lobject` specified).

- `[/box=r] (type RECT)`: Specifies a box for selection by box mode. Select by box operation depends on **Select→Selection Modes** in the same way as the **Select→Select by Box** command.
- `[/point=pt] (type POINT)`: Specifies a point for selection by point mode. The point must lay on the boundary of the specified object or on the centerline if object is wire.
- `[/segment={pt1, pt2}] (type SEQUENCE)`: Defines an edge or centerline of a specified object by a sequence of two points.
- `[/select | /reselect | /deselect]`: Cluster of named arguments that modify the default selection mode defined by the **Select→Selection Modes** command.
  - `/select` adds new stretch segments.
  - `/reselect` deselects previously selected segments and selects new ones.
  - `/deselect` deselects picked segments.

Example 1

```
__expert_setup.selectboxmode = SBM_INSIDE;
stretch select /box=(rect_create(0, 0, 5, 6)) /reselect;
```

Selects the segments laying inside specified box.

Example 2

```
stretch select (box 0 0 10 10 /layer = "Layer1" /find /get)
/point=(point_create(4, 10)) /select;
```

Selects the segment of the specified box object.

STRETCH RUN

Performs stretch operation.

Format

```
STRETCH RUN  shift_x shift_y
```

Arguments

```
shift_x(type DOUBLE)
shift_x(type DOUBLE)
```

Defines the stretch shift vector `v(shift_x, shift_y)`.

UNDO

Performs one step of undo.

Format

```
UNDO
```

17.11: Search Commands and Procedures

FIND OBJECTS

Looks for layout objects of specified types according to various criteria.

Format

```
objs = (FIND OBJECTS [object_type]
[/criteria = {criteria}]
[/search_rect = rect]
[/cross | /inside]
[/selected]
[/visible]
[/hierarchy]
[/seq_output]
```

Arguments

```
object_type (predefined constants)
```

Specifies the type of object for search. The following predefined constants are used to specify object type:

```
SEARCH_ANY_OBJECT (all layout objects)
SEARCH_ANY_SHAPE (primitive layout objects)
SEARCH_BOX
SEARCH_ELLIPSE
SEARCH_REGION
SEARCH_WIRE
SEARCH_TEXT
SEARCH_INSTANCE (including instance arrays)
SEARCH_ARRAY
```

If the argument `object_type` is not specified, then a search for all objects (`SEARCH_ANY_OBJECT`) will be performed.

Named Arguments

```
[/criteria = {criteria}] (type SEQUENCE)
```

`{criteria}={criterion`$_1$`,criterion`$_2$`,...,criterion`$_n$`}` is a sequence of search criterion (type `SEARCHCRITERION`). Use "SEARCH_CRITERION_CREATE" on page 17-111 for creating search criterion.

```
[/search_rect = rect] (type RECT)
```

Specifies the search region. If the named argument is not specified, then a search in the whole cell will be performed.

Switches

`[/cross | /inside]`

Cluster of switches to locate. Valid only if the search region is specified.

- `/cross` switch means that search will be performed among layout objects intersecting the search region.
- `/inside` switch means that search will be executed inside search region.

`[/selected]`

If specified, then a search will be performed among selected objects only.

`[/visible]`

If specified, then a search will be performed among visible objects only.

`[/hierarchy]`

If specified, then a search through the cell hierarchy will be performed.

`[/seq_output]`

If specified, then search results will be returned rather in the form of a sequence of layout objects (type `SEQUENCE`) then as a container of layout objects (type `OBJECTCONTAINER`).

Return Value

`objs (type OBJECTCONTAINER, SEQUENCE)`

Searched layout objects that satisfy all search criteria in the form of container of layout objects (type `OBJECTCONTAINER`) or sequence of layout objects (type `SEQUENCE`) depending from `/seq_output` switch. See the description for the `OBJECTCONTAINER` data type in Section 17.20.5: “OBJECTCONTAINER Type”.

Example 1

```
>criteria = {search_criterion_create(OA_LAYER, “Metal”, EQ)}
>objs = (FIND OBJECTS (SEARCH_BOX) /criteria = (criteria) /visible -
>/hierarchy)
```

Hierarchical search for visible box objects in the current cell on layer “`Metal`” is performed. The result is in the form of a container of layout objects.

Example 2

```
>criteria = {search_criterion_create(OA_LAYER, “Metal”, EQ)}
>objs = (FIND OBJECTS (SEARCH_WIRE) /criteria = (criteria) /seq_output)
```

Search for wire objects in the current cell on layer “`Metal`” is performed. The result is in the form of a sequence of layout objects.

SEARCH

To search for objects of specified types in the specified area according to various criteria.

Note: This command is obsolete. Use `‘FIND OBJECTS’` command with the switch `/seq_output` specified.

17.12: Modify Object Commands

These sections describes commands that modify layout objects specified by variables of LAYOUTOBJECT data type.

17.12.1: Common Arguments and Named Arguments

Note: If both argument 'lobject' and switch '/selected' are not specified than selected objects will be modified. If both are specified than switch '/selected' is ignored during command execution.

Arguments

```
[lobject] (type LAYOUTOBJECT, OBJECTCONTAINER, SEQUENCE)
```

The layout object is represented by the variable of LAYOUTOBJECT data type, a container of layout objects (type OBJECTCONTAINER), or a sequence of layout objects (type SEQUENCE). Use commands for creating objects (Section 17.8: "Commands for Creating Objects") with switch /get or FIND OBJECTS command (Section 17.11: "Search Commands and Procedures") to obtain layout objects.

Named Arguments

```
[/selected]
```

If specified, than selected objects will be modified.

```
[/layer = layer_name] (type STRING)
```

The name of a layer. Changes the layer of a primitive object if specified. Not applicable for instance layout objects.

```
[/property = {}] (type SEQUENCE)
[/xpos = value] (type INTEGER, FLOAT, DOUBLE)
```

The value of the horizontal position. Sets the new horizontal position for object if specified. Not applicable for layout objects with shape polygon or wire.

```
[/ypos = value] (type INTEGER, FLOAT, DOUBLE)
```

The value of a vertical position. Sets the vertical position for object if specified. Not applicable for layout objects with shape polygon or wire.

```
[/net = net_name]  (type STRING)
```

The name of a layer. Adds object to the specified net. Empty net name is used to remove object from the net the last one belongs.

```
[/pin = TRUE | FALSE]  (type BOOLEAN)
```

If specified, it changes the object status in the net (applicable only for layout objects included to the net).

17.12.2: Modify Commands

MODIFY OBJECT

Changes the common attributes for specified objects.

Format

```
MODIFY OBJECT [lobject] [/selected] [/xpos = value] [/ypos = value] [/
property = {}]
```

See Section 17.12.1: “Common Arguments and Named Arguments” for description of common arguments.

MODIFY PRIMITIVE

Changes the common attributes for primitive layout objects.

Format

```
MODIFY PRIMITIVE [lobject] [/selected] [/xpos = value] [/ypos = value]
[/property = {}] [/layer = layer_name] [/net = net_name] [/pin]
```

See Section 17.12.1: “Common Arguments and Named Arguments” for description of common arguments.

MODIFY BOX

Changes the box specific layout object attributes.

Format

```
MODIFY BOX [lobject] [/selected]  [/xpos = value] [/ypos = value] [/property
= {}]
[/layer = layer_name] [/net = net_name] [/pin]
[/xsize = value] [/ysize = value]
```

Named Arguments

```
[/xsize = value] (type INTEGER, FLOAT, DOUBLE)
```

The value of box horizontal size or ellipse horizontal radius. Sets the new horizontal size if specified.

```
[/ysize = value] (type INTEGER, FLOAT, DOUBLE)
```

The value of box vertical size or ellipse vertical radius. Sets the new vertical size if specified.

See Section 17.12.1: “Common Arguments and Named Arguments” for description of common arguments.

MODIFY TEXT

Changes the text specific layout object attributes.

Format

```
MODIFY TEXT [lobject] [/selected]  [/xpos = value] [/ypos = value]
[/property = {}]
[/layer = layer_name] [/net = net_name] [/pin]
[/text = string]
[/height = value] [/width = value]
[/alignstyle =]
[/slope = value] [/slant = ] [/mirror =]
```

Named Arguments

```
[/text = string] (type STRING)
```

Single-line text. Sets a specific single-line text label for text object if specified.

```
[/height = value] (type INTEGER, FLOAT, DOUBLE)
```

The value of a text height. Sets the specific height for text if specified.

```
[/width = value] (type INTEGER, FLOAT, DOUBLE)
```

The value of text width. Sets a specific width for text if specified.

```
[/alignstyle =] (predefined constants)
```

Sets the new text alignment style if specified. Text alignment with respect to its origin is defined by the following predefined constants:

```
TA_TOPLEFT, TA_TOPCENTER, TA_TOPRIGHT,
TA_MIDDLELEFT, TA_MIDDLELEFT, TA_MIDDLERIGHT,
TA_BOTTOMLEFT, TA_BOTTOMCENTER, TA_BOTTOMRIGHT.
[/slope = value] (type INTEGER, FLOAT, DOUBLE)
```

The value of the text slope in degree. Sets the new text slope angle if specified.

```
[/slant = ] (type BOOLEAN)
```

Changes text slant mode if specified: text will be slanted if named argument is set to TRUE, and slant mode will be OFF if named argument is set to FALSE.

```
[/mirror =] (type BOOLEAN)
```

Changes text mirroring mode if specified: text will be mirrored if named argument is set to TRUE, and mirroring mode will be OFF if the named argument is set to FALSE.

See Section 17.12.1: "Common Arguments and Named Arguments" for description of common arguments.

MODIFY WIRE

Changes the wire specific layout object attributes.

Format

```
MODIFY WIRE  [lobject] [/selected]  [/property = {}]
[/layer = layer_name] [/net = net_name] [/pin]
[/width = value]
[/endstyle = ] [/jointstyle =]
```

Arguments

```
[/endstyle = ] (predefined constants)
```

Sets the specified wire end style. Wire end style is defined by the following predefined constants:

```
DEFAULT_END, EXT_END, FLUSH_END, ROUND_JOINT.

[/jointstyle =] (predefined constants)
```

Sets the specified wire joint style. Wire joint style is defined by the following predefined constants:

```
DEFAULT_JOINT, EXT_JOINT, FILL_JOINT, ROUND_JOINT, MITER_JOINT.
```

See Section 17.12.1: "Common Arguments and Named Arguments" for description of common arguments.

MODIFY INSTANCE

Changes the instance specific layout object attributes.

Format

```
MODIFY INSTANCE  [lobject] [/selected]  [/xpos = value] [/ypos = value] [/
property = {}]
[/cell = cell_name] [/lib = lib_logname]
```

```
[/scale = value] [/rotation = value] [/mirror = value]
[/net = net_name] [/pin] [/conn = connectivity]
[/row = value] [/col = value]
[/xcol = value] [/ycol = value]
[/xrow = value] [/yrow = value]
```

Named Arguments

`[/cell = cell_name] (type STRING)`

The name of a cell (no full cell name). Replaces a cell instance object with the instance of the new cell.

`[/lib = lib_logname] (type STRING)`

The logical name of the library. Replaces a cell instance object with the instance of the cell in the new library.

`[/scale = value] (type INTEGER, FLOAT, DOUBLE)`

The scale value. Sets the new scale factor for an instance object if specified.

`[/rotation = value] (type INTEGER, FLOAT, DOUBLE)`

The value of a rotation angle in degree. Sets the new rotation angle for an instance object if specified.

`[/mirror = value] (predefined constants)`

Sets the specified mirroring type for an instance object. Mirroring type is defined by the following predefined constants:

```
MT_NONE, MT_HORIZONTAL, MT_VERTICAL
```

`[/conn = connectivity] (type SEQUENCE)`

Updates connectivity for the specified instances. See description of 'INSTANCE' command on page 17-49 for named argument `/conn` .

`[/col = value] (type INTEGER)`

Sets new number of columns in a cell instance array if specified.

`[/row = value] (type INTEGER)`

Sets new number of rows in a cell instance array if specified.

`[/xcol = value] (type INTEGER, FLOAT, DOUBLE)`

Sets new `X` shift for a cell instance array columns if specified.

`[/ycol = value] (type INTEGER, FLOAT, DOUBLE)`

Sets new `Y` shift for a cell instance array columns if specified.

`[/xrow = value] (type INTEGER, FLOAT, DOUBLE)`

Sets new `X` shift for a cell instance array rows if specified.

`[/yrow = value] (type INTEGER, FLOAT, DOUBLE)`

Sets new `Y` shift for a cell instance array rows if specified.

See Section 17.12.1: "Common Arguments and Named Arguments" for description of common arguments.

CONVERT POLYGON

Converts a primitive layout object into polygon.

Format

```
CONVERT POLYGON lobject
```

Arguments

```
lobject (type LAYOUTOBJECT, SEQUENCE)
```

See Section 17.12.1: "Common Arguments and Named Arguments" for argument description.

SPLIT ARRAY

Splits an existing cell array into rows, columns or separate instances.

Format

```
SPLIT ARRAY lobject [/splcol] [/splrow]
```

Arguments

```
lobject (type LAYOUTOBJECT, SEQUENCE)
```

See Section 17.12.1: "Common Arguments and Named Arguments" for argument description.

Switches

```
[/splcol]
```

If specified, then the cell instance array is divided into columns.

```
[/splrow]
```

If specified, then the cell instance array is divided into rows.

If both switches are specified, then the cell instance array is divided into separate instances.

17.13: Tools Commands

CLIP

Saves a rectangular area of the edited cell as a new cell.

Format

```
CLIP x y xs ys cell_name [/out | /in]
```

Arguments

```
x (type INTEGER, FLOAT, DOUBLE)
y (type INTEGER, FLOAT, DOUBLE)
xs (type INTEGER, FLOAT, DOUBLE)
ys (type INTEGER, FLOAT, DOUBLE)
```

x, y, x_s, y_s are defined as a rectangular area, where (x, y) is a left bottom corner and x_s and y_s are dimensions of the rectangle.

```
cell_name (type STRING)
```

The name of the new cell.

Switches

```
[/out | /in]
```

- /out is used to perform "Clip out" operation. It saves all geometry of the edited cell bordered with specified rectangular area into the new cell with name specified by cell_name argument.

- `/in` is used to perform “Clip in” operation. In addition to “Clip out” the following actions are performed. The part of original cell bounded by specified rectangle is cut through whole hierarchy.

Example

```
>clip 0 0 10 20 “cell_in” /in
```

Objects bound by a rectangle ((0, 0), (10,20)) are replaced by the new cell named “cell_in”.

CUT LINE

Cuts primitive objects by polyline. This command is subject to options set by the command `CUT MODE`. See “CUT MODE” on page 17-28 for more information.

Format

`CUT LINE {`x_1`,`y_1`, `x_2`,`y_2`,…,`x_n`,`y_n`}`

Arguments

`{,`y_1`, `x_2`,`y_2`,…,`x_n`,`y_n`} (type SEQUENCE)`

The sequence of vertices of a polyline. You may substitute any (x_i,y_i) pair by variable of `POINT` type.

Arguments

```
>cut mode /selected
>cut line {0,0, 3,3}
```

Selected objects, intersected by specified polyline, are cut.

DETACH TEXT

Detaches all attached texts from selected objects.

Format

```
DETACH TEXT
```

Example

```
>detach text
```

MERGE SELECTION

Merges (performs union) of the selected objects in the edited cell.

Format

```
MERGE SELECTION
```

Example

```
>merge selection
```

POLYGONIZE

Converts wires and ellipses to polygonal regions.

Format

`POLYGONIZE [/selected | /wire | /ellipse]`

Switches

`[/selected | /wire | /ellipse]`

Cluster of switches defines scope of objects to be converted to polygons (*/selected* is a default switch).

- /selected switch means that selected wires and ellipses will be converted.
- /wire switch means that all wires in the current project will be converted to polygons.
- /ellipse switch means that all ellipses in the current project will be converted to polygons.

Example

```
>polygonize /wire;
```

All wires in the current project are converted to polygons.

RESIZE SELECTION

Expands or shrinks the selected objects of the edited cell.

Format

```
RESIZE SELECTION value [/over | /under] [/savegrid]
```

Arguments

```
size (type INTEGER, FLOAT, DOUBLE)
```

The amount of resizing in the current user units.

Switches

[/over | /under]

- /over defines the operation "Oversize" (default operation).
- /under defines the operation "Undersize" operation (default operation).

[/savegrid]

If specified then region vertices will remain on the grid.

Example

```
>resize selection 2 /over /grid
```

SCALE

Rescales the set of selected objects or whole layout according to scale parameters.

Format

```
SCALE x y factor
[/total | /dimension | /placement]
[/lefttop | /centertop | /righttop |
/leftmid | /centermid | /rightmid |
/leftbottom | /centerbottom | /rightbottom | /nearest]
[/selected]
```

Arguments

```
x  (type INTEGER, FLOAT, DOUBLE)
y  (type INTEGER, FLOAT, DOUBLE)
```

(x, y) is a center of scale operation ().

```
factor  (type INTEGER, FLOAT, DOUBLE)
```

Specifies a scale value.

Switches

```
[/total | /dimension | /placement]
```

Cluster of switches defines scaling type. /total is the default switch.

```
[/lefttop | /centertop | /righttop |
/leftmid | /centermid | /rightmid |
/leftbottom | /centerbottom | /rightbottom | /nearest]
```

Cluster of switches defines the location of scaling reference point (not valid for /total (scaling both dimension and placement) scaling type). /centerbottom is the default switch.

```
/selected
```

If specified, then only selected objects will be scaled. The whole layout will be scaled otherwise.

SNAP GRID

Snaps selected objects or whole layout to the specified grid.

Format

```
SNAP GRID [/snap | /minimal | /visual | /custom]
[/stepx = value] [/stepy = value] [/shiftx = value] [/shifty = value]
[/nearest | /northeast | /outwards | /inwards]
[/refpointx = value] [/refpointy = value]
[/savesquare] [/save45] [/cutspikes]
[/selected]
```

Named Arguments

```
[/stepx = ] (type INTEGER, FLOAT, DOUBLE)
```

Specifies “custom” grid step in X direction.

```
[/stepy =] (type INTEGER, FLOAT, DOUBLE)
```

Specifies “custom” grid step in Y direction.

```
[/shiftx =] (type INTEGER, FLOAT, DOUBLE)
```

Specifies the shift of “custom” grid step in X direction with respect to origin.

```
[/shifty =] (type INTEGER, FLOAT, DOUBLE)
```

Specifies the shift of “custom” grid step in Y direction with respect to origin.

Note: Named arguments /stepx, /stepy, /shiftx, and /shifty are used only for “custom grid” (switch /custom is specified).

```
[/refpointx =] (type INTEGER, FLOAT, DOUBLE)
[/refpointy =] (type INTEGER, FLOAT, DOUBLE)
```

Note: Named arguments /refpointx and /refpointy are used only for shifting modes specified by switches /inwards or /outwards.

Use for /outwards and /inwards switches.

Switches

```
[/snap | /minimal | /visual | /custom]
```

Cluster of switches to specify the grid. If no switches in the cluster are specified, the /snap switch is used.

- /snap, /minimal and /visual define "snap", "minimal" and "visual" grids correspondingly.
- /custom switch means that grid is defined by named arguments: /stepx, /stepy, /shifxy and /shifty.

[/nearest | /northeast | /outwards | /inwards]

Cluster of switches to specify shifting direction for the off-grid points. If no switches in the cluster is specified, the /nearest switch is used.

- /nearest switch means that offgrid points will be shifted to the closest grid point.
- /northeast switch means that offgrid points will be shifted to the grid point with both coordinate increase.
- /outwards switch means that offgrid points will be shifted to the closest grid point in the direction outwards, with respect to a specified reference point by the named arguments /refpointx and /refpointy.
- /inwards switch means that offgrid points will be shifted to the closest grid point in the direction inwards, with respect to a specified reference point by the named arguments /refpointx and /refpointy.

[/savesquare]

If specified.

[/save45]

Preserves angles up to 45 degrees.

[/cutspikes]

Allows you to save 45 degree geometry.

[/selected]

If specified, then only selected objects will be shifted to the grid point; the whole layout will be shifted otherwise.

TOOLS CUTBYVERTEX

Cuts large polygon and wires (by means of vertices count) into smaller ones with a specified number of vertices

Format

```
TOOLS CUTBYVERTEX [/all | /regions /wires] [/selected | /project]
                  [/regionvertices=value] [wirevertices=value]
```

Arguments

- [/all | /regions /wires]: Specifies the type of objects to be cut:
  - /all (default) means that wires and regions will be cut;
  - /regions is used to cut regions;
  - /wires is used to cut wiress;
- [/selected | /project]: Specifies the operation scope:
  - /selected (default) means that only selected polygons and (or) wires will be cut.
  - /project is used to cut all polygons and (or) wires in the current cell library.

- `[/regionvertices=value] (type Integer)`: Specifies the number of vertices for cut regions. If the argument is not present, the default max number of wire vertices is used (see the menu command Tools→Cut by Vertices Count→Cut by Vertices Count Setup).
- `[/wirevertices=value] (type Integer)`: Specifies the number of vertices for cut wires. If the argument is not present, the default max number for wire vertices is used (see the menu command **Tools→Cut by Vertices Count→Cut by Vertices Count Setup**).

Example

```
tools cutbyvertex /wires /selected /wirevertices=4;
```

Cuts all selected wires in the active cell view into wires with a number of no more than 4 vertices.

TOOLS SPLIT

Splits compound objects (arrays, P-wires, and multipaths) into separate objects: instances, wires and boxes.

Format

```
TOOLS SPLIT [/all | /arrays /mpaths | /pwires] [/selected | /project]
```

Arguments

- `[/all | /arrays /mpaths | /pwires]`: Specifies the type of objects to be split:
  - `/all` (default) means that arrays, P-wires and multipaths will be split;
  - `/arrays` is used to split regions (array splitting mode: rows, columns, all) is defined by the menu command **Tools→Split→Array Split Mode**);
  - `/mpaths` is used to split maltipaths;
  - `/pwires` is used to split P-wires.
- `[/selected | /project]`: Specifies the operation scope:
  - `/selected` (default) means that only objects will be split.
  - `/project` is used to split all specified objects in the current cell library.

  `[/regionvertices=value] (type Integer)`

Example

```
tools split /pwires /selected;
```

Splits all selected P-wires in the active cell view.

WIRING

Performs wiring between specified source and target contacts.

Format

```
wire = (WIRING [source] [target] [rarea=] [layer=] [width=] [space=])
```

or

```
WIRING [source] [target] [rarea=] [layer=] [width=] [space=]
```

Arguments

`source (type POINT, SEQUENCE)`

Specifies source contact or contacts.

`target (type POINT, SEQUENCE)`

Specifies target contact or contacts.

Named Arguments

```
[/rarea] (type RECT)
```

Specifies the routing area. If the argument is omitted, the routing area is defined automatically.

```
[/layer] (type STRING)
```

The layer name. If not specified, the active layer is used.

```
[/width] (type INTEGER, FLOAT, DOUBLE)
```

The wire width. If not specified, the wire width for specified layer will be applied.

```
[space=] (type INTEGER, FLOAT, DOUBLE)
```

The spacing from the wire to the obstacles. If not specified, the wire width value will be applied.

Return Value

The wire handle (type `LAYOUTOBJECT`) if wiring is succeeded, and NIL otherwise.

Example

```
ptsource = point_create(5, 3);
pttarget = point_create(-3, -3);
wiring (ptsource) (pttarget);
```

WC_ADD_CONTACT

Adds source or target contact for wiring. Afterwards, the `'WIRING'` command can be used without `'source'` and `'target'` arguments.

Format

```
WC_ADD_CONTACT [cnt] /source|/target
```

Arguments

```
[cnt] (type POINT or LAYOUTOBJECT).
```

Specifies a point or an object node for adding to the source or target contact's set. If this argument is omitted, then objects from the preliminarily selected node are added. Note, if the new contact already exists in the corresponding contacts set, then the last one is excluded from the specified contact group.

Switches

```
/source | /target
```

Specifies the contact group the new one will be added to: `/source` means sources and `/target` means targets, respectively.

Example

```
wc_add_contact (point_create(12, -1)) /source; ! adds point contact to the
source contact group
wc_add_contact (box 17 -1.5 5 21 /layer="Metal1" /find /get) /source; ! adds
object contact
node_pick(point_create(-0.5, -9), "Metal1"); ! selects node
wc_add_contact /target; !add node to contact group
```

WC_REMOVE_CONTACTS

Removes source and target contacts from the wiring context.

Format

```
WC_REMOVE_CONTACTS
```

Arguments

None.

Switches

None.

17.14: Hierarchy Commands

CELL CREATE

See “CELL CREATE” on page 17-8.

EXPLODE

Explodes cell instances in the edited cell up one level of the cell hierarchy.

Format

```
EXPLODE [/all | /selected]
```

Switches

```
[/all | /selected]
```

- `/all` switch means that all instances and arrays in the edited cell are exploded.
- `/selected` (default switch) switch means that selected instances and arrays in the edited cell are exploded.

Example

```
>explode /all
```

All instances (arrays) in the edited cell are exploded.

FLATTEN

Flattens (completely removes hierarchy) the edited cell into a file.

Format

```
FLATTEN project_name
```

Arguments

```
project_name (type STRING)
```

The name of the `eld` file to write the flattened cell.

Example

```
>flatten “mux_flat”
```

Flattened image of the current cell is saved into the ELD project file “`mux_flat.eld`”.

REPLACE INSTANCES

Replaces the instances of a cell with the instances of another cell.

Format

```
REPLACE INSTANCES cell_name new_cell_name [/cell | /project]
```

Arguments

```
cell_name (type STRING)
```

Name of cell for replacing with.

```
new_cell_name (type STRING)
```

Name of new cell for replacing by.

Switches

```
[/cell | /project]
```

- `/cell` (default) switch means that instances will be replaced in the current cell.
- `/project` switch means that instances will be replaced in the whole project.

Example

```
>replace instances "inverter" "inverter2" /project
```

Instances of the cell "`inverter`" will be replaced by instances of "`inverter2`" in the whole project.

REPLACE SELECTED

Replaces the selected cell instances by instances of the another cell.

Format

```
REPLACE SELECTED cell_name
```

Arguments

```
cell_name (type STRING)
```

Name of the new cell.

Example

```
>replace selected "inverter2"
```

Selected instances of the current cell will be replaced by instances of the cell "`inverter2`".

17.15: Edit-in-place Commands

EIP COMPLETE

Completes the editing in place in cell instance (array).

Format

```
EIP POP [/save | /saveas | /explode | /reject] [/newname=cell_name]
```

Named Arguments

```
[/newname =] (type STRING)
```

Name of the cell the changes will be saved to. Valid for `/saveas` switch only.

Switches

```
[/save | /saveas | /explode | /reject]
```

- `/save` applies changes to the edited cell.
- `/saveas` allows you to save changes to the new cell with the name defined by the named argument `/newname`. The instances of a new cell will replace the instances of the edited cell. The edited cell will then be deleted.
- `/explode` inserts the instances of the edited cell into its parent cell. The edited cell will then be deleted.
- `/reject` discards changes in the edited cell.

Example

```
>eip complete /saveas /newname = “nand2_1”;
```

Completes an editing in place session with saving edited cell with new name “nand_2”.

EIP EXIT

Terminates the EIP session.

Format

```
EIP EXIT
```

Example

```
>eip exit
```

EIP POP

Goes out one level in the EIP instance chain.

Format

```
EIP POP
```

Example

```
>eip pop
```

EIP PUSH

Selects a cell instance/array to edit in place.

Format

```
EIP PUSH instance [/colnum =]  [/rownum =]
```

Arguments

```
instance (type LAYOUTOBJECT)
```

An instance (array) object to be edited in place.

Named Arguments

```
[/colnum =] (type INTEGER)
[/rownum =] (type INTEGER)
```

Named arguments /colnum and /rownum are used to choose a specific element of instance array to edit in place. /colnum is a number of column, and /rownum is a number of row in the specified instance array. Default value for both named arguments is 0.

Example

```
>arr = (array 0 0 /cell = "mux1" /col = 4 /row = 1 /xdeltacol = 88 -
>/ydeltacol = 0 /xdeltarow = 0 /ydeltarow = 64 /find /get;
>eip push (arr) /colnum = 0 /rownum = 0;
```

Starts EIP session for element (0, 0) of 1x4 instance array of cell “mux1”.

17.16: Connectivity Commands

NET_CREATE

Creates a new net in the current cell.

Format

```
NET_CREATE net_name [/internal | /external | /global | /type=value]
```

Arguments

```
net_name (type STRING)
```

Name of the net.

Named Arguments

```
[/internal | /external | /global | /type=value] (predefined switches and
constants)
```

Switch `/internal` is used for creating internal nets. Internal nets are served for internal interconnections and not available for cell instance connectivity.

Switch `/external` means that an external net is created. External nets are used while creating cell instance connectivity.

Switch `/global` means that a global net is created. Global nets have not to be added to the each cell. Once created global nets are available in the all cells.

One of the following predefined constants can be used to specify type of the net:

```
NT_INTERNAL,
NT_EXTERNAL,
NT_GLOBAL.
```

If neither the named argument `/type` nor the predefined switch are not specified, then an external net is created.

NET_DELETE

Deletes the specified net from the current cell.

Format

```
NET_DELETE net_name
```

Arguments

```
net_name (type STRING)
```

Name of the net to be deleted.

NET_ADD_OBJ

Adds layout object to the specified net in the current cell. If object is not specified, then selected objects are added.

Format

```
NET_ADD_DELETE net_name [lobject]
```

Arguments

```
net_name (type STRING)
```

Name of the net.

```
[lobject] (type LAYOUTOBJECT)
```

Layout object to be added to the net (optional argument).

NET_SUB_OBJ

Subtracts the layout object from the specified net in the current cell. If the object is not specified, then selected objects are subtracted.

Format

```
NET_SUB_DELETE net_name [lobject]
```

Arguments

```
net_name (type STRING)
```

Name of the net.

```
[lobject] (type LAYOUTOBJECT)
```

Layout object to be subtracted the net (optional argument).

NET_ADD_CONN

Connects the instance cell net to the net of the current cell.

Format

```
NET_ADD_CONN net_name instance_name instance_cell_net_name
```

Arguments

```
net_name (type STRING)
```

Name of the net.

```
instance_name (type STRING)
```

Name of the instance object.

```
instance_cell_net_name (type STRING)
```

Name of the net in the instance cell.

NET_SUB_CONN

Disconnects the instance cell net from the net of the current cell.

Format

```
NET_SUB_CONN net_name instance_name instance_cell_net_name
```

Arguments

```
net_name (type STRING)
```

Name of the net.

```
instance_name (type STRING)
```

Name of the instance object.

```
instance_cell_net_name (type STRING)
```

Name of the net in the instance cell.

17.17: Verification Commands

DRC COMMAND

Execute DRC command in the current cell.

Format

```
DRC COMMAND command
[/xpos = value] [/ypos = value] [/xsize = value] [/ysize = value]
[/limit = value]
[/selection | /cell | /cellwise | /whole
[/hierarchical | /flat]
[/log]
```

Arguments

```
command (type STRING)
```

Specifies DRC command. It must end with a semicolon and be written in double quotes, see the example.

Named Arguments

```
[/xpos =] (type INTEGER, FLOAT, DOUBLE)
[/ypos =] (type INTEGER, FLOAT, DOUBLE)
[/xsize =] (type INTEGER, FLOAT, DOUBLE)
[/ysize =] (type INTEGER, FLOAT, DOUBLE)
```

`/xpos`, `/ypos`, `/xsize` and `/ysize` define the region to run the check: by the coordinates of the lower left corner and the sizes of the box. The command `DRC SCRIPT` without these arguments will run the check in the whole area of the current cell.

```
[/limit =] (type INTEGER)
```

Sets limit on the number of reported errors (100,000 is default value).

Switches

```
[/selection | /cell | /cellwise | /whole]
```

Cluster of switches that define the scope of the layout object that will be checked while running DRC.

- `/selection` switch means that scope is a set of preliminary selected objects.
- `/cell` switch means that cell instance objects will not be taken.
- `/cellwise` switch means that all primitives from current cell and cells instanced directly or indirectly will be processed separately.
- `/whole` switch that DRC will be executed on a flattened image of the current cell.

```
[/hierarchical | /flat]
```

Cluster of switches that define the mode of the error report.

```
[/log]
```

If selected then log file will be generated.

Example

```
>str_comm =  "OR: layer1=POLY, layer2=CONT, layerr=PCO;"
>drc command (str_comm) /limit=1000 /hierarchical
```

DRC_IMPORT_ERRORS

Imports the DRC errors stored either in the GUARDIAN DRC error database, or in the other systems such as Mentor Graphics' Calibre or Cadence's Dracula error reports.

Format

```
DRC_IMPORT_ERRORS file_name [/guardian | /dracula | /calibre]
```

Arguments

- `file_name (type STRING)`: Specifies the pathname of a DRC error database file.

Switches

```
[/guardian | /dracula | /calibre]
```

Defines the type of DRC error database.

- `/guardian` (default switch) means an import from GUARDIAN DRC's error database.
- `/dracula` means an import from Cadence's Dracula database.
- `/calibre` means an import from Mentor Graphics' Calibre text error report.

DRC SCRIPT

Executes DRC script from file in the current cell.

Format

```
DRC SCRIPT pathname
[/xpos = value] [/ypos = value] [/xsize = value] [/ysize = value]
[/limit = value]
[/selection | /cell | /cellwise | /whole
[/hierarchical | /flat]
[/log]
```

Arguments

pathname (type STRING)

The name of the DRC script file.

See "DRC COMMAND" on page 17-87 for an explanation of other named arguments and switches.

EXTRACT CONNECTIVITY

Extracts node connectivity data for the current cell view. It corresponds to the menu command **Verification→Node Probing→Reextract Connectivity**.

Format

```
EXTRACT CONNECTIVITY
```

Example

```
>extract connectivity
```

This xi-command is useful as a preparation for node probing for large layouts.

NODE_PICK( )

Picks a node for further operations.

Syntax

```
ret = NODE_PICK (pt, layer_name)
```

Parameters

```
pt (type POINT)
```

Specifies a point inside of the probed object.

```
layer_name (type STRING)
```

Specifies the layer of the probed object. If the empty string is specified, then the probed object is assumed to be in the current layer.

Return Value

```
ret (type BOOLEAN)
```

TRUE if the pick node succeeded, and FALSE otherwise.

NODE_PICK_BY_NAME

Picks a node by its name.

Syntax

```
ret = NODE_PICK_BY_NAME(node_name)
```

Parameter

- `node_name (type String)`: Specifies the name of node to be picked.

Return Value

```
ret (type BOOLEAN)
```

`TRUE` if the pick was succeeded, and `FALSE` otherwise.

NODE_GET_NAME( )

Returns the name of the picked node.

Syntax

```
node_name = NODE_GET_NAME()
```

Return Value

```
node_name (type STRING)
```

Name of the picked node. If no nodes picked then 'NIL' returned.

NODE_COPY( )

Copies picked node to the new cell.

Syntax

```
ret = NODE_COPY(cell_name)
```

Parameters

```
cell_name (type STRING)
```

Specifies cell name where selected node will be placed.

Return Value

```
ret (type BOOLEAN)
```

TRUE if the operation succeeded, and FALSE otherwise (e.g., if `cell_name` already exists)

17.18: Library/User Management Commands and Procedures

17.18.1: Library Setup Commands

LIBRARY ACTIVATE

Activates library for the current project.

Format

```
LIBRARY ACTIVATE lib_logname
```

Arguments

```
lib_logname (type STRING)
```

The logical name of the library.

LIBRARY DEACTIVATE

Deactivates the library for the current project. You cannot deactivate a library if some of its cell are used in the opened projects.

Format

```
LIBRARY DEACTIVATE lib_logname
```

Arguments

```
lib_logname (type STRING)
```

The logical name of the library.

LIBRARY NAME REMOVE

Deletes the library from EXPERT'S workspace.

Format

```
LIBRARY NAME REMOVE lib_logname [/all]
```

Arguments

```
lib_logname (type STRING)
```

The logical name of the library.

Switches

```
[/all]
```

If the switch `/all` is specified, then all libraries will be deleted from EXPERT'S workspace.

LIBRARY NAME SET

Adds a library into EXPERT'S workspace.

Format

```
LIBRARY NAME SET lib_logname lib_pathname
```

Arguments

```
lib_logname (type STRING)
```

The logical name of the library.

```
lib_pathname (type STRING)
```

The full or short name of the file for the library. If the short file name is given, the library will be searched in the current directory.

LIBRARY TYPE

Sets library type.

Format

```
LIBRARY TYPE lib_pathname [/exclusive | /readonly]
```

Arguments

```
lib_pathname (type STRING)
```

The full or short name of the file for the library. If the short file name is given, the library will be searched in the current directory.

Switches

```
[/exclusive | /readonly]
```

- `/exclusive` (default switch) makes the library an exclusive library: only one user can edit the library at any moment.
- `/readonly` (default switch) makes library as a read-only: cells cannot be edited.

17.18.2: Library Management Procedures

GET_ACTIVATED_LIBRARY_LIST

Gets the sequence of logical names of activated libraries from the EXPERT'S workspace.

Syntax

```
libs = GET_ACTIVATED_LIBRARY_LIST()
```

Return Value

Sequence of logical names of activated libraries from EXPERT'S workspace (type `SEQUENCE`).

GET_COMPUTER_NAME

Gets computer name.

Syntax

```
comp = GET_COMPUTER_NAME()
```

Return Value

Name of computer (type `STRING`).

GET_LIBRARY_LIST

Gets the sequence of library logical names from the EXPERT'S workspace.

Syntax

```
libs = GET_LIBRARY_LIST()
```

Return Value

Sequence of logical names from EXPERT'S workspace (type `SEQUENCE`).

LIBRARY_ACTIVATED

Returns TRUE if library is activated, and FALSE otherwise.

Syntax

```
ret = LIBRARY_ACTIVATED(lib_logname)
```

Parameters

```
lib_logname (type STRING)
```

The logical name of the library

Return Value

TRUE if library activated, and FALSE otherwise (type BOOLEAN).

Example

```
>display(library_activated("A"));
```

LIBRARY_EXISTS

Returns TRUE if library exists, and FALSE otherwise.

Syntax

```
ret = LIBRARY_EXISTS(lib_logname)
```

Parameters

```
lib_logname (type STRING)
```

The logical name of the library.

Return Value

TRUE if library exists, and FALSE otherwise.

Example

```
>display(library_exists("A"));
```

17.18.3: Cell Management Commands

CELL CHECKOUT

Checks the cell out from the shared library for editing.

Format

```
CELL CHECKOUT [cell_name]
```

Arguments

```
cell_name (type STRING)
```

The name of the cell to be checked out. If no argument is given, the command will be applied to the current cell, if it is from the shared library.

CELL CHECKIN

Checks the cell in the shared library if it was checked out by the current user.

Format

```
CELL CHECKIN [cell_name]
```

Arguments

```
cell_name (type STRING)
```

The name of the cell to be checked in. If no argument is given, the command will be applied to the current cell, if it is from the shared library.

CELL CANCEL CHECKOUT

Discards changes in the checked out cell.

Format

```
CELL CANCEL CHECKOUT [cell_name]
```

Arguments

```
cell_name (type STRING)
```

The name of the cell. If no argument is given, the command will be applied to the current cell, if it is from the shared library.

17.19: Miscellaneous

FORM_CREATE

Creates a data entry form to initialize the XI shell variables.

Syntax

```
vars = FORM_CREATE(var_params, form_params)
```

Parameters

```
var_params (type SEQUENCE)
```

The description of the variables. Each element of the `var_params` is a sequence containing the initial value and the description of the variable to be initialized. Initial values define the data type and input field type. `INTEGER`, `DOUBLE`, `STRING`, and `BOOLEAN` data types are allowed. Use tab-separated string for the drop-down selection list.

```
form_params (type SEQUENCE)
```

Sequence containing parameters of the dialog input form. Currently, only one parameter (the title of the entry form dialog) is used.

Return Value

```
var_params (type SEQUENCE).
```

Values of variables.

Example

This example creates the entry form with title "test input form" with specified initial values (see Figure 17-17-1). The sequence "`vars`" contains initialized variables.

.

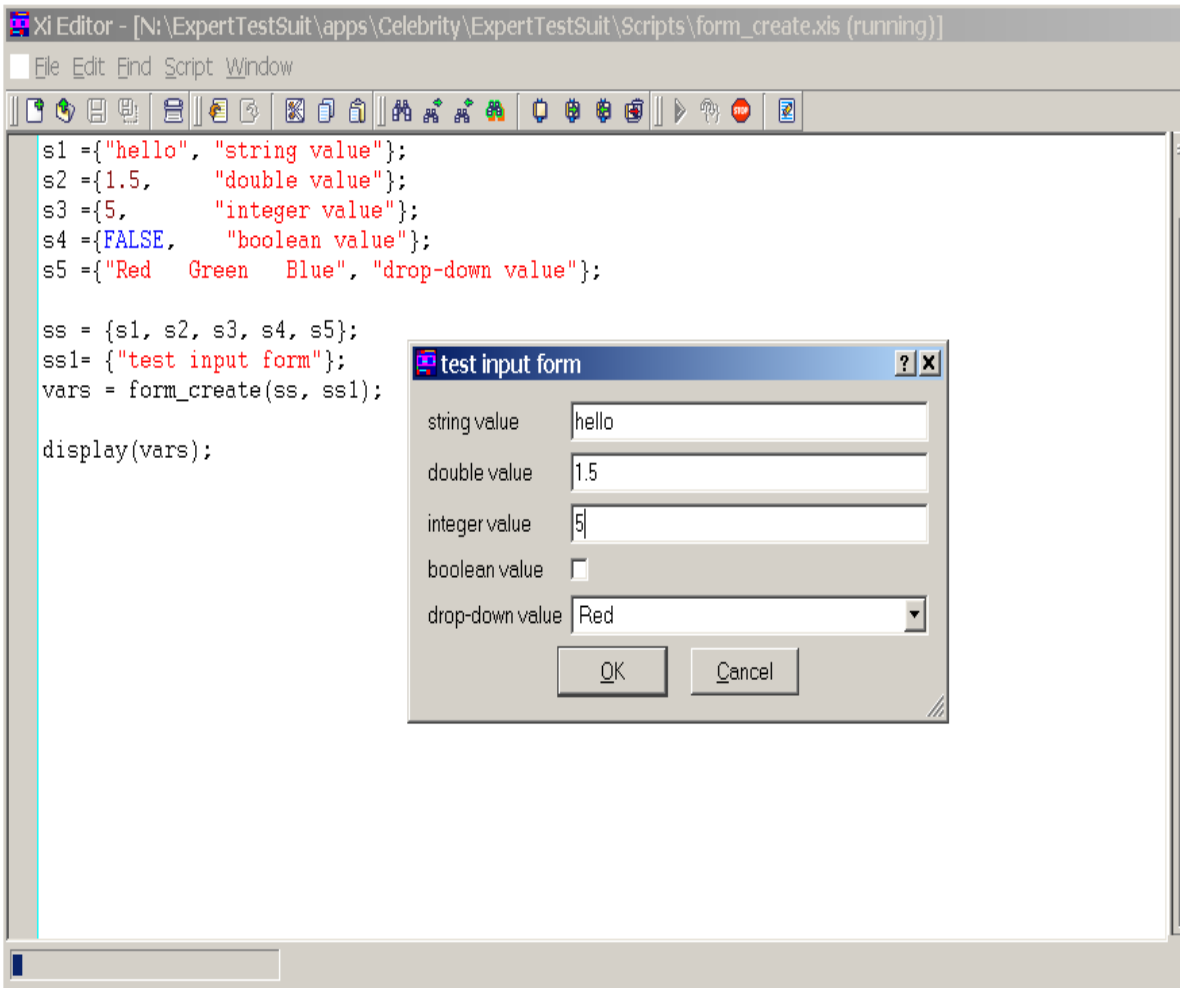


Figure 17-1: Calling Create Form Window from Xi-script

FILE_EXISTS

Returns True if the file or directory exists, and False otherwise.

Syntax

```
ret = FILE_EXISTS(pathname)
```

Parameter

`pathname (type STRING)`

File or directory path name

Return Value

`ret (type BOOLEAN)`

True if the file or directory exists, and False otherwise.

Example

```
display(file_exists("d:/GDSII/mux4.gds"));
```

FILE_SELECTOR

Invokes the **File Selector** dialog box to input a file name, see Figure 17-17-2.

Syntax

```
fname = FILE_SELECTOR()
```

Return Value

```
fname (type STRING)
```

The string with the file pathname.

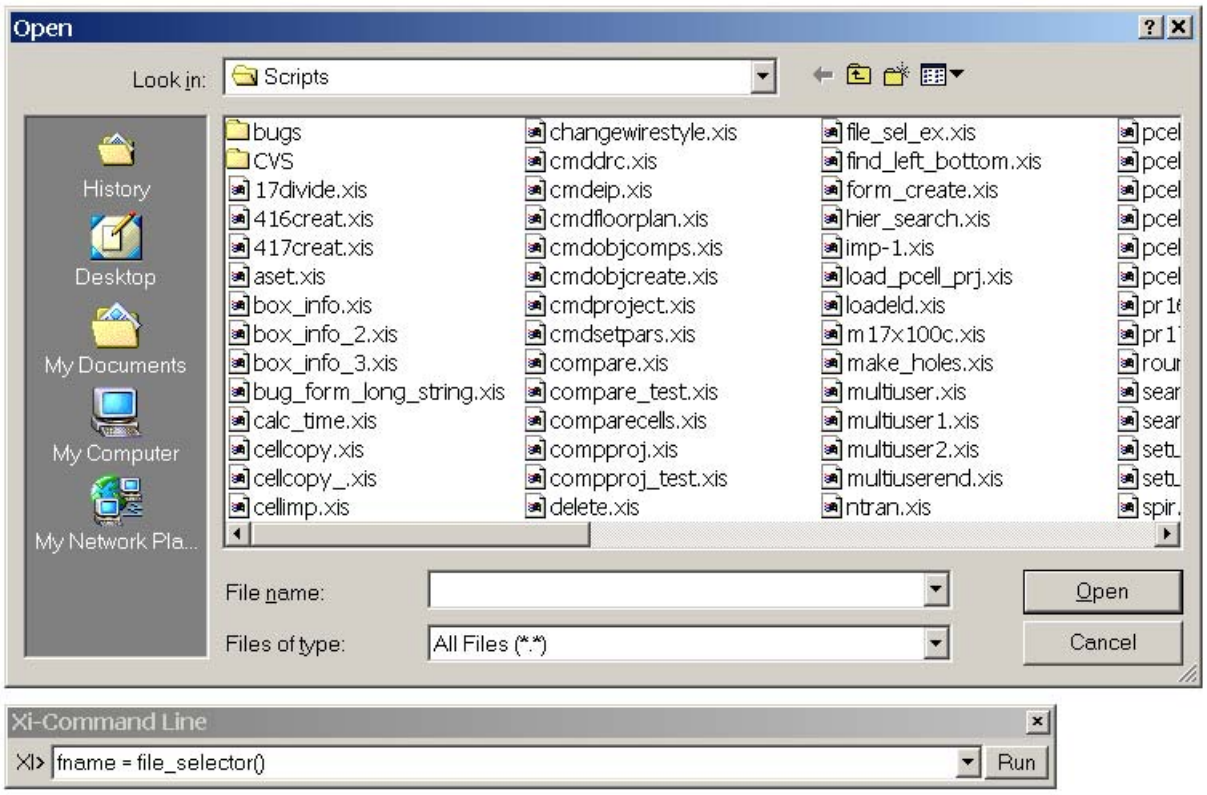


Figure 17-2: Calling File Selector Dialog Window from Xi-script

FILE_SELECTOR_EX

Invokes the **File Selector** dialog box with extended capabilities.

Syntax

```
fname = FILE_SELECTOR_EX(params)
```

Parameters

```
params (type SEQUENCE)
```

Sequence of file dialog attributes has the form:

```
params = {title, [openfile, def_ext, init_dir, {file_filters}]}
```

where:

```
title (type STRING)
```

Title of the file dialog.

`openfile (type BOOLEAN)`

If TRUE then “Open file” dialog is created; “Save as file” dialog otherwise.

`def_ext (type STRING)`

The default file name extension. This extension will be automatically added to the file name if you do not type an extension.

`init_dir (type STRING)`

The path name of the initial directory.

`file_filters (type SEQUENCE)`

A sequence of string pairs in the form:

`description, file_types`

that describe files to be opened or saved (see the example).

Return Value

`fname (type STRING)`

The string with the file pathname.

Example

```
>display(file_selector_ex({"LISA Script files", true, "*.xis", -
>"d:/test/scripts", { "LISA Scripts (*.xis)", "*.xis", -
>"Plain text", "*.txt"}}))
```

Creates an “Open file” dialog box for LISA Scripts files with extension “`*.xis`” or “`*.txt`“ in the directory “`d:/test/scripts`“ and displays the selected file name (see Figure 17-17-3)

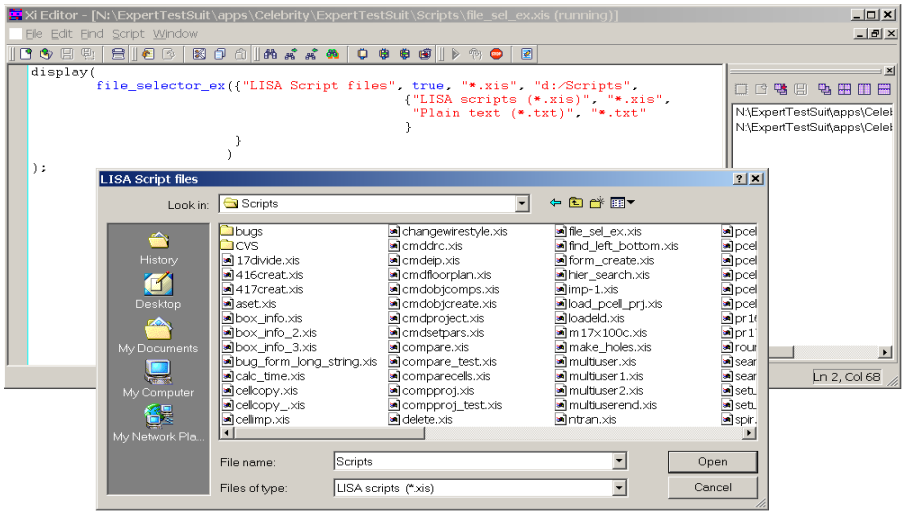


Figure 17-3: File Dialog with Extended Capabilities

GET_ENVIRONMENT_STRING

Retrieves the value of the specified variable from the environment block.

Note: This command is obsolete. Use the command `'GETENV'` (see Chapter 10: "General Library" in the "LISA user's manual".

GET_POINT

Returns mouse position.

Syntax

```
pt=GET_POINT ()
```

Parameters

None.

Return Value

```
pt (type POINT)
```

Coordinate values in current measurement units for the cursor position. It takes into an account the active reference point.

If it is called from xi-script, then it expects a mouse click in the layout.

If `get_point()is` called, rather than clicking the mouse, hit the **Esc** key. The function will return a NIL. Remember that when `get_point()` expects input of a point, you can still use the following operations:

- viewing (panning, zooming)
- Ruler
- Pick Layer
- operations in the Layer bar

If any other EXPERT's interactive operations are selected (box creation, stretch, etc.), then NIL will be returned.

Example 1

```
>p1 = get_point()
>p2 = get_point()
>box (p1.x) (p1.y) (p2.x-p1.x) (p2.y-p1.y)
```

Example 2

```
>pseq = {}
>LOOP BEGIN -
>p2 = get_point() -
>if (p2. EQL NIL) then (LEAVE LOOP); -
>seq_add_last(pseq, p2); -
>END;
>if (pseq.size GTR 2) then (polygon (seq));
```

GET_POINT_EX

Returns the mouse position depending on the current direction snap mode.

Syntax

```
pt=GET_POINT _EX({params})
```

Parameters

```
params (type SEQUENCE)
```

Sequence of parameters has the form:

```
params = {[prev_point]}
```

where:

```
prev_point (type POINT)
```

Specifies the point on the current layout window. The selection of the next point depends on the current direction snap mode with respect to the `prev_point` point. If this sequence is empty, i.e., `get_point_ex({})` syntax is used, the command works as "`get_point()`".

Return Value

```
pt (type POINT)
```

Coordinate values in current measurement units for the cursor position. It takes into an account the active reference point.

Example

```
>! orthogonal cursor movement mode selected
>p1 = point_create(0, 0)
>p2 = get_point_ex({p1});
```

GET_RECT

Returns the rectangle defined by a mouse click at the two opposite corners in the current layout window.

Syntax

```
r = GET_RECT()
```

Parameters

None.

Return Value

```
r (type RECT).
```

The components of the rectangle are in the current measurement units and take into account the active reference point. If the **Esc** key is pressed during the operation, the procedure returns NIL value.

Example

```
>r1 = get_rect()
```

IS_DOUBLE

Validates the input string for conversion to variable of type `DOUBLE`.

Syntax

```
ret = IS_DOUBLE(string)
```

Parameters

```
string (type STRING).
```

String to test.

Return Value

```
ret (type BOOLEAN).
```

TRUE if conversion is possible, and FALSE otherwise.

Example 1

```
>display(is_double("3e40")
>TRUE
```

Example 2

```
>display(is_double("abc")
>FALSE
```

IS_FLOAT

Validates the input string for conversion to variable of type `FLOAT`.

Syntax

```
ret = IS_FLOAT(string)
```

Parameters

```
string (type STRING).
```

String to test.

Return Value

```
ret (type BOOLEAN).
```

TRUE if conversion is possible and FALSE otherwise.

Example 1

```
>display(is_float("3.5")
>TRUE
```

Example 2

```
>display(is_float("3.a")
>FALSE
```

IS_INTEGER

Validates the input string for conversion to variable of type `INTEGER`.

Syntax

```
ret = IS_INTEGER(string)
```

Parameters

```
string (type STRING).
```

String to test.

Return Value

```
ret (type BOOLEAN).
```

TRUE if conversion is possible, and FALSE otherwise.

Example 1

```
>display(is_integer(“3“)
>TRUE
```

Example 2

```
>display(is_integer(“3.5“)
>FALSE
```

MESSAGE_BOX

It is the **“Yes”/”No”** dialog box with extended capabilities.

Syntax

```
ret = MESSAGE_BOX(string, params)
```

Parameters

```
string (type STRING)
```

Text string to be displayed. Unlike `“YES_NO_DLG”`, this text is displayed in a scrollable window, and therefore has no limitation on the length of the message.

```
params (type SEQUENCE)
```

The sequence of parameters that control the appearance (title, buttons, return values, etc.) of the message box. In the current implementation `message_box()` recognizes only the following cases:

```
{} (empty parameter sequence)
```

In this case it produces an ordinary **Yes/No** dialog.

```
{“”} (nonempty parameter sequence)
```

It produces a **Yes/No** dialog with scrollable window for long messages.

Return Value

```
ret (type BOOLEAN)
```

True if the **Yes** button was clicked, and False otherwise.

OUTSIDE_XIS

Delays currently running xi-script and switches to EXPERT. The **Tools→Xi-script→Return** command resumes xi-script execution.

Syntax

```
OUTSIDE_XIS()
```

Parameters

None.

Return Value

None.

POINT_INSIDE

Tests whether the point is strictly inside of a layout object. “Strictly” means the boundary is not inside.

Syntax

```
ret = POINT_INSIDE(pt, lobject)
```

Parameters

```
pt (type POINT)
```

Point to be examined.

```
lobject (type LAYOUTOBJECT)
```

Layout object to be tested.

Return Value

```
ret (type BOOLEAN)
```

TRUE if the point is inside the layout object, and FALSE if otherwise.

The "inside" of an object is defined consistently only for box, region, floorplan block, floorplan pad.

For wire and circle/ellipse it depends on the current settings in **Setup→Technology→General→Shapes**.

For cell instance/array it is determined by position and bounding box; the latter depends on the current contents of the placed subcell.

For floorplan pin in depends on the current setting for pin size.

For text (label) it depends on the current drawing implementation for text font and its size.

Example

```
> b1 = (box 0 0 5 5 /get);
> display(point_inside(point@"2 2", b1);
```

POINT_TRANSFORM_GLOBAL

Transforms the point to a global coordinate system concerning the local coordinate system of a cell in the edit-in-place mode. If the EIP session is not initiated then the initial point is returned.

Syntax

```
pt_global = POINT_TRANSFORM_GLOBAL(pt)
```

Parameters

```
pt (type POINT)
```

Point in the local coordinate system.

Return Value

```
pt_global (type POINT)
```

Point in the global coordinate system.

POINT_TRANSFORM_LOCAL

Transforms point to global coordinate system concerning local coordinate system of cell in the edit-in-place mode. If the EIP session is not initiated then initial point is returned.

Syntax

```
pt_global = POINT_TRANSFORM_GLOBAL(pt)
```

Parameters

```
pt (type POINT)
```

Point in the global coordinate system.

Return Value

```
pt_global (type POINT)
```

Point in the local coordinate system.

SPAWN

Executes a new program.

Note: This command is obsolete. Use the command `'SH'` (see Chapter 10: "General Library" in the "LISA user's manual".

Example 1

```
> SH "notepad";
```

Starts WINDOWS'S NOTEPAD and waits until program is finished.

Example 2

```
> SH "xemacs";
```

Starts Unix's xemacs and waits until program is finished.

Example 3

```
> SH "start notepad";
```

Executes WINDOWS's NOTEPAD concurrently with an EXPERT process.

Example 4

```
> SH "xemacs &";
```

Executes Unix's xemacs concurrently with EXPERT process.

YES_NO_DLG

Yes/No dialog box.

Syntax

```
ret = YES_NO_DLG(string)
```

Parameters

```
string
```

Text string to be displayed in the dialog box.

Return Value

```
ret (type BOOLEAN)
```

True if **Yes** button was selected, and False if **No** button was selected.

17.20: EXPERT/LISA Data Types

This section describes additional predefined LISA data types that are defined to execute specific EXPERT actions.

17.20.1: POINT Type

```
Type-object constant: POINT
```

The `POINT` type represents a coordinate, which is a pair of doubles.

Table 17-1: Point Basic Operation

Basic operation	Operator	Second operand	Behavior
`equal`	`EQL`	`POINT`	Returns TRUE if two points are equal (have the same x and y component); returns FALSE otherwise.
`not equal`	`NEQ`	`POINT`	Returns TRUE if two points are not equal (have the different x and y component); returns FALSE otherwise.
`to string`	`&`	`POINT`	Returns a string containing point components.

Table 17-2: Point Components

Component	Access	Value Type	Behavior
`X`	`get`	`DOUBLE`	Returns the x coordinate.
`X`	`put`	`DOUBLE`	Sets the x coordinate.
`Y`	`get`	`DOUBLE`	Returns the y coordinate.
`Y`	`put`	`DOUBLE`	Sets the y coordinate.
`TYPE`	`get`	`TYPE`	Returns the RECT type constant.

17.20.2: RECT Type

Type-object constant: RECT

The RECT type represents a rectangle defined by the coordinates of the lower left corner, and the dimensions of the rectangle, which are pairs of doubles.

Table 17-3: Rect Basic Operation

Basic operation	Operator	Second operand	Behavior
equal	EQL	RECT	Returns TRUE if two rects have the same left corner and dimensions; returns FALSE otherwise.
not equal	NEQ	RECT	Returns TRUE if two rects do not have the same left corner and dimensions; returns FALSE otherwise.
to_string	&	none	Returns a string containing the parameters of rect.

Table 17-4: Rect Components

Component	Access	Value Type	Behavior
XPOS	get	DOUBLE	Returns the x coordinate of left corner.
XPOS	put	DOUBLE	Sets the x coordinate of left corner.
YPOS	get	DOUBLE	Returns the y coordinate of left corner.
YPOS	put	DOUBLE	Sets the y coordinate of left corner.
XSIZE	get	DOUBLE	Returns the value of “x” dimension.
XSIZE	put	DOUBLE	Sets the value of “x” dimension.
YSIZE	get	DOUBLE	Returns the value of “y” dimension.
YSIZE	put	DOUBLE	Sets the value of “y” dimension.
TYPE	get	TYPE	Returns the RECT type constant.

17.20.3: LAYOUTOBJECT Type

```
Type-object constant: LAYOUTOBJECT
```

The LAYOUTOBJECT type is the handle of a layout object of an EXPERT database. To obtain a variable of LAYOUTOBJECT type, use the functions from Section 17.11: "Search Commands and Procedures".

Table 17-5: LayoutObject Basic Operation

Basic Operation	Operator	Second Operand	Behavior
equal	EQL	LAYOUTOBJECT	Returns TRUE if two objects represent the same layout object of EXPERT data base; returns FALSE otherwise.
not equal	NEQ	LAYOUTOBJECT	Returns TRUE if two objects do not represent the same layout object of EXPERT data base; returns FALSE otherwise.
to_string	&	none	Returns a string containing the parameters of the layout object.

Table 17-6: LayoutObject Common Components

Component	Access	Value Type	Behavior
AREA	get	DOUBLE	Returns an area of the layout object.
BBOX	get	RECT	Returns a bounding box of layout object.
CELLOWNER	get	STRING	Return name of cell.
LAYER	get	STRING	Returns name of layer (not valid for instance layout objects).
LAYER	put	STRING	Places objects on the new layer (not valid for instance layout objects).
PERIMETER	get	DOUBLE	Returns a perimeter of layout object.
POSITION	get	POINT	Returns a position of layout object (not valid for polygons and wires).
POSITION	put	POINT	Sets a new position of layout object (not valid for polygons and wires).
PROPERTY	get	ATTRIBUTE_SET	Returns the user-defined properties if the object has ones, and NIL otherwise.
PROPERTY	put	ATTRIBUTE_SET	Sets the user-defined properties.
SELECTED	get	BOOLEAN	Returns TRUE if object is selected and FALSE otherwise.
SELECTED	put	BOOLEAN	Sets or resets selecting.
SHAPE	get	STRING	Returns the shape of the object ("box", "ellipse", "polygon", "wire", "text", "instance", "array").
TYPE	get	TYPE	Returns the LAYOUTOBJECT type constant.

Property names in the EXPERT database are case sensitive, therefore you must use case-sensitive attribute sets (see Section 17.2: “Remarks on Command Syntax and Descriptions” for more information). If you assign a new attribute set for the layout object, the old attribute set is deleted.

Example

```
>inst1 = (instance 39.5 -4 /rotate = 90 /cell = "nand2" /find /get);
>property = inst1.property;
>if (property EQL NIL) then (property = aset_create_cs());
>property["NodeName"] = "Node1"; ! add new property attribute
>inst1.property = property;      ! update properties for the instance
>display((inst1.property)["NodeName"])
```

Table 17-7: LayoutObject Box (Ellipse) Components

Component	Access	Value type	Behavior
SIZE	get	POINT	Returns a size (radius) of box (ellipse).
SIZE	put	POINT	Sets a size (radius) of box (ellipse).

Table 17-8: LayoutObject Polygon Components

Component	Access	Value type	Behavior
COORDS	get	SEQUENCE	Returns a sequence of vertex (type POINT) of polygon.
COORDS	put	SEQUENCE	Sets a new vertices (type POINT) of polygon.

Table 17-9: LayoutObject Text Components

Component	Access	Value type	Behavior
ALIGN	get	STRING	Returns a “align style” of text.
ALIGN	put	STRING	Sets a new “align style” of text.
HEIGHT	get	DOUBLE	Returns a height of text characters.
HEIGHT	put	DOUBLE	Sets a new height of text characters.
LABEL	get	STRING	Returns a label of text.
LABEL	put	STRING	Sets a new label of text.
MIRROR	get	BOOLEAN	Returns TRUE if text is mirrored and FALSE otherwise.
MIRROR	put	BOOLEAN	Sets or resets mirroring status.
SLANT	get	BOOLEAN	Returns TRUE if text is slanted and FALSE otherwise.
SLANT	put	BOOLEAN	Sets or resets a slanted text.

Table 17-9: LayoutObject Text Components

Component	Access	Value type	Behavior
SLOPE	get	DOUBLE	Returns a slope of text line in degrees.
SLOPE	put	DOUBLE	Sets a new slope of text line in degrees.
WIDTH	get	DOUBLE	Returns a width of text characters.
WIDTH	put	DOUBLE	Sets a new width of text characters.

Table 17-10: LayoutObject Wire Components

Component	Access	Value type	Behavior
COORDS	get	SEQUENCE	Returns a sequence of vertex (type POINT) of wire.
COORDS	put	SEQUENCE	Sets a new vertices (type POINT) of wire.
ENDSTYLE	get	STRING	Returns a "end style" of wire.
ENDSTYLE	put	STRING	Sets a new "end style" of wire.
JOINTSTYLE	get	STRING	Returns a "joint style" of wire.
JOINTSTYLE	put	STRING	Sets a new "joint style" of wire.
LENGTH	get	DOUBLE	Returns a length of wire.
WIDTH	get	DOUBLE	Returns a width of wire.
WIDTH	put	DOUBLE	Sets a new width of wire.

Table 17-11: LayoutObject Instance Components

Component	Access	Value type	Behavior
CELL	get	STRING	Returns name of cell (valid for instances (arrays) only).
CELL	put	STRING	Sets new cell.
COLDELTA	get	POINT	Returns a column delta.
COLDELTA	put	POINT	Sets a new column delta.
COLS	get	INTEGER	Returns a number of columns in array.
COLS	put	INTEGER	Sets a new number of columns in array.
LIBRARY	get	STRING	Returns a logical name of library for instanced cell.
MIRROR	get	BOOLEAN	Returns TRUE if instance is mirrored and FALSE otherwise.
MIRROR	put	BOOLEAN	Sets or resets mirroring status.
PARAMS	get	SEQUENCE	Returns parameters of P-cell instance.

Table 17-11: LayoutObject Instance Components

Component	Access	Value type	Behavior
PARAMS	put	SEQUENCE	Sets new parameters for P-cell instance.
PARAMNAMES	get	SEQUENCE	Returns parameter's names of P-cell instance.
ROTATION	get	DOUBLE	Returns a rotation angle of instance.
ROTATION	put	DOUBLE	Sets a new rotation angle of instance.
ROWDELTA	get	POINT	Returns a row delta.
ROWDELTA	put	POINT	Sets a new row delta.
ROWS	get	INTEGER	Returns a number of rows in array.
ROWS	put	INTEGER	Sets a new number of rows in array.
SCALE	get	DOUBLE	Returns a scaling factor.
SCALE	put	DOUBLE	Sets a new scaling factor.

17.20.4: SEARCHCRITERION Type

```
Type-object constant: SEARCHCRITERION
```

The SEARCHCRITERION type represents a criterion that finds objects with specific attributes using the search function. To obtain a variable of SEARCHCRITERION type, use the procedure SEARCH_CRITERION_CREATE (see "SEARCH_CRITERION_CREATE" on page 17-111).

Table 17-12: SearchCriterion Operations

Basic Operation	Operator	Second Operand	Behavior
equal	EQL	SEARCHCRITERION	Returns TRUE if two objects represent the same search criterion; returns FALSE otherwise.
not equal	NEQ	SEARCHCRITERION	Returns TRUE if two objects do not represent the same search criterion; returns FALSE otherwise.

Table 17-13: SearchCriterion Components

Component	Access	Value type	Behavior
TYPE	get	TYPE	Returns the SEARCHCRITERION type constant.

17.20.5: OBJECTCONTAINER Type

```
Type-object constant: OBJECTCONTAINER
```

Layout object containers are containers for accumulating of huge amount of layout objects (type LAYOUTOBJECT) with effective memory management. Using this data type instead of predefined SEQUENCE data type increase performance greatly when dialing with large sets of layout objects, for example, result of hierarchical search command (see Section 17.11: "Search Commands and Procedures").

Table 17-14: OBJECTCONTAINTER Operations

Basic Operation	Operator	Second Operand	Behavior
equal	EQL	OBJECTCONTAINER	Returns TRUE if two objects represent the same layout object container, returns FALSE otherwise.
not equal	NEQ	OBJECTCONTAINER	Returns TRUE if two objects do not represent the same layout object container, returns FALSE otherwise.

Table 17-15: OBJECTCONTAINTER Components

Component	Access	Value type	Behavior
CURRENT	get	LAYOUTOBJECT	Returns the current layout object.
CURRENT	put	LAYOUTOBJECT	Replaces the current layout object.
FIRST	get	LAYOUTOBJECT	Returns the first layout object.
LAST	get	LAYOUTOBJECT	Returns the last layout object.
PREV	get	LAYOUTOBJECT	Returns the previous layout object.
NEXT	get	LAYOUTOBJECT	Returns the next layout object.
SIZE	get	INTEGER	Returns the number of layout objects.
TYPE	get	TYPE	Returns the SEARCHGROUP type constant.

17.20.6: EXPERT/LISA Data Type Procedures

POINT_CREATE

Creates a point data object from the coordinates.

Syntax

```
pt = POINT_CREATE(x, y)
```

Parameters

```
x (type INTEGER, FLOAT, DOUBLE)
y (type INTEGER, FLOAT, DOUBLE)
```

Return Value

```
pt (type POINT)
```

Example

```
> pt = point_create(5, 4.5)
```

Creates a new point (5, 4.5)

RECT_CREATE

Creates a rectangle data object.

Syntax

```
r = RECT_CREATE(xpos, ypos, xsize, ysize)
```

Parameters

x_{pos} (type INTEGER, FLOAT, DOUBLE)

y_{pos} (type INTEGER, FLOAT, DOUBLE)

x_{size} (type INTEGER, FLOAT, DOUBLE)

y_{size} (type INTEGER, FLOAT, DOUBLE)

Return Value

```
r (type RECT)
```

Example

```
>r = rect_create(0, 0, 5, 7)
```

Creates a new rectangle (5, 4.5) with position (0, 0) and dimensions 5 and 7 along the x and y directions respectively.

SEARCH_CRITERION_CREATE

Creates SEARCHCRITERION data object.

Syntax

```
criterion = SEARCH_CRITERION_CREATE(obj_attr, value, condition)
```

Parameters

```
obj_attr
```

Specifies the layout object attribute to be compared with a specified value. See predefined constants below.

```
value (type STRING, INT, FLOAT, DOUBLE, ATTRIBUTE)
```

Specifies the value of layout object attribute for comparison with.

```
condition
```

An operator used for comparing the value of a layout object attribute and the value of a search parameter. Use predefined constants:

- EQ: Equal to
- NE: Not equal to
- GT: Greater than
- GE: Greater than or equal to
- LT: Less than
- LE: Less than or equal to
- SUBSTR: Substring

- FILTER: Filter

Return Value

```
criterion (type SEARCHCRITERION)
```

Search criterion objects for using in the SEARCH procedure (see Section 17.11: "Search Commands and Procedures").

Table 17-16: Layout Object Attributes (SEARCH_ANY_OBJECT)

Attribute	Value Data Type	Description
OA_PROPERTY	ATTRIBUTE	User-defined property.
OA_PROPERTY_INT	ATTRIBUTE	User-defined property as integer.
OA_PROPERTY_FLOAT	ATTRIBUTE	User-defined property as float.
OA_PROPERTY_BOOLEAN	ATTRIBUTE	User-defined property as boolean.

User-defined properties in the search criteria are described by ATTRIBUTE type values, where names of property correspond to names of attribute, and the value of property correspond the value of attribute. Case sensitive ATTRIBUTE should be used because EXPERT property name is case sensitive. Property value is stored in the EXPERT data base as a string. Use the predefined constants OA_PROPERTY_INT, OA_PROPERY_FLOAT and OA_PROPERTY_BOOLEAN if you want to process the property value as integer, float and boolean value correspondingly.

Example 1

```
>property = attr_create_cs("path width", "0.5");
>prop_crit = search_criterion_create(OA_PROPERTY_FLOAT, property, LE)
```

Creates a search criterion to find objects with properties "path width" less than or equal to 0.5

Example 2

```
>property = attr_create_cs("width", "");
>prop_crit = search_criterion_create(AO_PROPERTY, property, EQ)
```

Creates a search criterion to find objects with properties "width" with any values of given property.

Table 17-17: Primitive Layout Object Attributes (SEARCH_ANY_SHAPE)

Attribute	Value Data Type	Description
OA_LAYER	STRING	Layer name.

Example 3

```
>layer_crit = search_criterion_create(OA_LAYER, "Met1", NE)
```

Creates a search criterion to find primitives that are not in layer "Met1".

Table 17-18: Box(Ellipse) Layout Object Attributes

Attribute	Value Data Type	Description
OA_POSX	DOUBLE	Object position X.
OA_POSY	DOUBLE	Object position Y.
OA_SIZEX	DOUBLE	Box size X (Ellipse radius X).
OA_SIZEY	DOUBLE	Box size Y (Ellipse radius Y).

Example 4

```
>px_crit = search_criterion_create(OA_POSX, 0.5, GT)
```

Creates a search criterion to find objects with horizontal position greater than 0.5 from the cell origin.

Example 5

```
>py_crit = search_criterion_create(OA_POSY, 0.5, GT)
```

Creates a search criterion to find objects with a vertical position greater than 0.5 from the cell origin

Example 5

```
>sx_crit = search_criterion_create(OA_SIZEX, 0.5, GT)
```

Creates a search criterion to find boxes with horizontal size (or ellipses with horizontal radius value) greater than 0.5.

Example 6

```
>sy_crit = search_criterion_create(OA_SIZEY, 0.5, GT)
```

Creates a search criterion to find boxes with vertical size (or ellipses with vertical radius value) greater than 0.5.

Table 17-19: Text Layout Object Attributes

Attribute	Value Data Type	Description
OA_POSX	DOUBLE	Object position X.
OA_POSY	DOUBLE	Object position Y.
OA_TALIGN	Predefined	Text alignment (see predefined constants on 17-74).
OA_TEXT	STRING	Text string.
OA_THEIGHT	DOUBLE	Height of character.
OA_TSLOPE	DOUBLE	Slope of the text line.
OA_TWIDTH	DOUBLE	Width of character.

Example 6

```
>tx_crit = search_criterion_create(OA_TEXT, “met”, SUBSTR)
```

Creates a search criterion to find texts which contain substring “met”.

Example 7

```
>th_crit = search_criterion_create(OA_THEIGHT, 1, GT)
```

Creates a search criterion to find texts with a height more than 1.

Example 8

```
>th_crit = search_criterion_create(OA_TWIDTH, 1.5, LT)
```

Creates a search criterion to find texts with a width that is less than 1.5.

Example 9

```
>ts_crit = search_criterion_create(OA_TSLOPE, 90, EQ)
```

Creates a search criterion to find texts with a slope of 90 degrees.

Example 10

```
>ts_crit = search_criterion_textalign(OA_TALIGN, TA_BOTTOMLEFT, EQ)
```

Creates a search criterion to find texts with a bottom left alignment.

Table 17-20: Wire Layout Object Attributes

Attribute	Value Data Type	Description
OA_WEND	Predefined	Wire end style (see predefined constants on 17-74).
OA_WJOINT	Predefined	Wire joint style (see predefined constants on 17-74).
OA_WWIDTH	Double	Wire width.

Example 11

```
>we_crit = search_criterion_create(OA_WEND, EXT_END, EQ)
```

Creates a search criterion to find wires with "Extend" end style.

Example 12

```
>wj_crit = search_criterion_create(OA_WJOINT, ROUND_JOINT, EQ)
```

Creates a search criterion to find wires with "Round" joint style.

Example 13

```
ww_crit = search_criterion_create(OA_WWIDTH, 0.3, EQ)
```

Creates a search criterion to find wires with a width of 0.3.

Table 17-21: Instance Layout Object Attributes

Attribute	Value Data Type	Description
OA_CELL	STRING	
OA_MIRROR	Predefined	Instance mirror status (see predefined constants on 17-75).

Table 17-21: Instance Layout Object Attributes

Attribute	Value Data Type	Description
OA_ROTATION	Double	Instance rotation angle.
OA_SCALE	Double	Instance scale factor.

Example 14

```
>rot_crit = search_criterion_create(OA_ROTATION, 90, EQ)
```

Creates a search criterion to find instances and arrays with rotations of 90 degrees.

Example 15

```
>mir_crit = search_criterion_create(OA_MIRROR, MT_VERTICAL, EQ)
```

Creates a search criterion to find instances and arrays with vertical mirroring.

Example 16

```
>sc_crit = search_criterion_create(OA_SCALE, 2, EQ)
```

Creates a search criterion to find instances and arrays with scale 2.

Example 17

```
>cell_crit = search_criterion_create(OA_CELL, “inverter”, EQ)
```

Creates a search criterion to find instances and arrays of a cell “inverter”.

17.20.7: EXPERT_SETUP Type

Type-object constant: EXPERT_SETUP

The EXPERT_SETUP data type represents the EXPERT preferences. The global variable __EXPERT_SETUP is available to change the EXPERT setup modes.

Table 16-x: EXPERT_SETUP Components

Component	Access	Value Type	Behavior
FORBID_SERLINTERSECT	get	BOOLEAN	Returns TRUE if creation of self intersected shapes forbidden; returns FALSE otherwise
FORBID_SERLINTERSECT	put	BOOLEAN	Allows (FALSE) or disallows (TRUE) creation of self intersected shapes
LPE_DERIVEDLAYERSCRIPT	put	STRING	Returns a pathname of external DRC script file
LPE_DERIVEDLAYERSCRIPT	get	STRING	Sets a pathname to the external DRC script file

LPE_DERIVEDLAYERTECH	put	BOOLEAN	Returns TRUE if current technology is used for layer derivation; return FALSE if external DRC script file is used
LPE_DERIVEDLAYERTECH	get	BOOLEAN	Defines how to calculate derived layers: current technology or external DRC script
REGIONMODE	get	INTEGER (predefined: RM_NORMAL, RM_MERGE, RM_HOLE, RM_HOLESELECTED)	Returns current Region mode for creation of shape object
REGIONMODE	put	INTEGER (predefined: SM_SELECT, SM_DESELECT, SM_RESELECT)	Sets the Region mode
SELECTBOXMODE	get	INTEGER (predefined: SBM_INSIDE, SBM_CROSS)	Returns current Select by Box mode
SELECTBOXMODE	put	INTEGER (predefined: SBM_INSIDE, SBM_CROSS)	Sets the Select by Box Mode
SELECTFILLMODE	get	INTEGER (predefined: SFM_BOUNDARY, SFM_SOLID)	Returns current Select Fill mode
SELECTFILLMODE	put	INTEGER (predefined: SFM_BOUNDARY, SFM_SOLID)	Sets the Select Fill mode
SELECTINSTANCES	get	BOOLEAN	Returns TRUE, if instances are selectable; returns FALSE otherwise
SELECTINSTANCES	put	BOOLEAN	Allows or disallows selecting instances
SELECTMODE	get	INTEGER (predefined: SM_SELECT, SM_DESELECT, SM_RESELECT)	Returns current Select Mode

SELECTMODE	put	INTEGER (predefined: SFM_BOUNDARY, SFM_SOLID)	Sets the Select Mode
SELECTPRIMITIVES	get	BOOLEAN	Returns TRUE, if primitives are selectable; returns FALSE otherwise
SELECTPRIMITIVES	put	BOOLEAN	Allows or disallows selecting primitives

Other components will be added in the order of their priority.

Example

```
>__expert_setup.lpe_derivedlayertech = false;
>__expert_setup.lpe_derivedlayerscript = "d:\drc\test.dsf";
```

The derived layers for HIPEX execution will be calculated using the DRC script `d:\drc\test.dsf` rather than current technology.

17.21: Creating P-cells command

P-cells in the EXPERT environment can be created either by using the P-cell graphical user interface, or by Xi-script using LISA and Xi predefined commands and procedures. Textual P-cells are more suitable for creation of complex designs because of using powerful features of LISA/Xi language that are impossible to generate automatically. The Xi command **DEFINE PCELL** passes the definition of P-cell script to the EXPERT data base in the form of -pcell master.

DEFINE PCELL

Defines a P-cell master.

Format

```
DEFINE PCELL pcell_name [/REPLACE]
                                [PARAMETER param_name /TYPE=type /DEFAULT=value
                                   ...]
                                   BODY BEGIN pcell_script END;
```

Arguments

```
pcell_name (type STRING)
```

The name of P-cell master. If you want to create or redefine P-cell in the activated library then full cell name should be used.

Switches

```
[/REPLACE]
```

Specifies that P-cell with specified name will be redefined if last one already exists. If the switch is not specified while a P-cell with this name exists, then a warning message is displayed and redefinition has not occurred.

PARAMETER Subcommand

Describes the parameters for P-cell definition.

Format

```
PARAMETER param_name /TYPE=type /DEFAULT=value
```

Arguments

```
pcell_name (type STRING)
```

The name of P-cell parameter.

Named Arguments

```
/TYPE=type (type TYPE)
```

Specifies the valid types that can be entered for the parameter. The following types are valid for P-cell parameters: `INTEGER`, `DOUBLE`, `BOOLEAN`, `STRING`.

```
/DEFAULT=value
```

Specifies the default parameter value. Type of value should coincide with type specified by named argument /type.

BODY Subcommand

Provides the implementation for the P-cell.

Format

```
BODY BEGIN pcell-script END
```

Arguments

```
pcell_name
```

Any LISA expression or block that defines the P-cell's implementation.

Example

```
DEFINE PCELL "ntran" /REPLACE
  PARAMETER gates /TYPE = (Integer) /DEFAULT = (1)
  PARAMETER width /TYPE = (Double) /DEFAULT = (3)
  PARAMETER length /TYPE = (Double) /DEFAULT = (1)
  PARAMETER layer_ndiff /TYPE = (String) /DEFAULT = ("Layer1")
  PARAMETER layer_poly /TYPE = (String) /DEFAULT = ("Layer2")
  PARAMETER layer_met1 /TYPE = (String) /DEFAULT = ("Layer3")
  PARAMETER layer_cont /TYPE = (String) /DEFAULT = ("Layer4")
  BODY BEGIN
  W          = width;
  W_def      = 3.;
  NGates     = gates;
  NGates_def = 1;
  L          = length;
  L_def      = 1.;
  SizeX_def  = 8.;      ! default ntran size x
  SizeY_def  = 3.;      ! default ntran size y
  Met_Poly_dist   = 1.; ! Distance between poly and met1
  Poly_Poly_dist  = 1.; ! Distance between poly if NGates > 1
  Cont_Cont_dist  = 1.; ! Distance between contacts
  Cont_size       = 1.; ! contact size
  Cont_NDiff_dist = 0.5;! Distance between contacts and ndiff
  Met_NDiff_offset= 0.5;
```

```
  Poly_NDiff_offset = 1.;

  IF (W LSS 2.) THEN (W = 2.);
  IF (L LSS 1.) THEN (L = 1.);
  IF (L GTR 5.) THEN (L = 5.);

  L_offset   = (L - L_def) / 2.; ! must be multiple by NGates
  W_offset   = (W - SizeY_def);
  StretchR   = (L + Poly_Poly_dist) * (NGates - 1) + (L - L_def);
!  StretchR = 0;

  ! ndiff
  layer (layer_ndiff);
  SizeX_ndiff = SizeX_def + StretchR;
  PosX_ndiff = -SizeX_def / 2. - L_offset;
  PosY_ndiff = -W / 2.;
  box (PosX_ndiff) (PosY_ndiff) (SizeX_ndiff) (W);

  ! metal
  StretchMetCont = StretchR - (L - L_def);
  layer (layer_met1);
  SizeX_met = SizeX_def / 2. - Met_NDiff_offset - Met_Poly_dist - L_def / 2.;
  SizeY_met = W - Met_NDiff_offset * 2;
  box (PosX_ndiff + Met_NDiff_offset) (PosY_ndiff + Met_NDiff_offset)
(SizeX_met) (SizeY_met);
  box (-PosX_ndiff - Met_NDiff_offset + StretchMetCont) (PosY_ndiff +
Met_NDiff_offset)
      (-SizeX_met) (SizeY_met);

  ! contacts
  layer (layer_cont);
  sizeY_all_conts = SizeY_met - Cont_NDiff_dist * 2;
  nConts = floor(sizeY_all_conts / (Cont_size + Cont_Cont_dist));
  rest = sizeY_all_conts - (Cont_size + Cont_Cont_dist) * nConts;
  nConts = nConts + floor(rest / Cont_size);
  PosX_cont = PosX_ndiff + Met_NDiff_offset + Cont_NDiff_dist;
  PosY_cont = W / 2 - Met_NDiff_offset - Cont_NDiff_dist -
              (sizeY_all_conts - nConts * (Cont_size + Cont_Cont_dist) +
Cont_Cont_dist)/ 2.;
  iCont = 1;
  LOOP BEGIN
    IF (iCont GTR nConts) THEN (LEAVE LOOP);
    box (PosX_cont) (PosY_cont) (Cont_size) (-Cont_size);
    box (-PosX_cont + StretchMetCont) (PosY_cont) (-Cont_size) (-Cont_size);
    PosY_cont = PosY_cont - Cont_size - Cont_Cont_dist;
```

```
      iCont = iCont + 1;
    END;
    ! poly
    layer (layer_poly);
    PosY_poly = PosY_ndiff - Poly_NDiff_offset;
    SizeY_poly = W + Poly_NDiff_offset * 2;
    PosX_poly = -L / 2;
    iGate = 1;
    LOOP BEGIN
      IF (iGate GTR NGates) THEN (LEAVE LOOP);
      box (PosX_poly) (PosY_poly) (L) (SizeY_poly);
      PosX_poly = PosX_poly + L + Poly_Poly_dist;
      iGate = iGate + 1;
    END;
  END;
```

17.22: Access to Xi-commands

17.22.1: Command Line Input

To input Xi-script editing commands interactively, it is necessary to switch the **Command Line** dock window on. It can be done in several ways:

- from the menu command, **View→Dock Windows→Xi-Command Line**, switch on by checking the corresponding line.

or

- you may assign a keyboard shortcut from the menu command, **Setup→Customize Keyboard**, to show/hide the command line bar. The visibility of control bars is under the category of "Show Control Bars" in the keyboard setup panel.

or

- by the menu command, **Tools→Xi Script→Xi-Command Line** and by a user-defined shortcut for it.

The history of all commands executed from the command line is in the drop-down list, which is opened by clicking the small button with the black triangle pointing down. You may select commands from history, and rerun them, possibly, after editing in the command line.

17.22.2: Xi Script Panel

The menu command, **Tools→Xi Script→Script Panel**, invokes the xi-script editor. Similar to the DRC script editor, it loads, saves, creates, edits, and executes xi-scripts. Files with xi-scripts have a `*.xis` extension.

Basically, there are two categories of scripts. Some of them define new EXPERT commands, e.g., for parametric creation of cells. The others directly execute some other editing actions.

Of course, a script file may both define and execute commands, but it is better to keep this distinction in mind.

After a new command is defined by running the corresponding xi-script, it may be executed from the command line. A xi-command may have any number of arguments (parameters).

17.22.3: Activity Log

The Activity Log is a record of all operations you executed in an xi-script format. The Activity Log is written if the corresponding option is selected in **Setup→Editor/Viewer→Data Safety**.

To replay the Activity Log, you must have the license for LISA language.

17.22.4: Cell→Export→Xi-script

This menu command writes the current cell into a file in xi-script format.

17.23: Command Quick Reference

Table 17-22:

Return type	Command/Procedure	Arguments/Named Arguments/Switches/ Parameters	Ref
	PROJECT NEW	proj_name [tech_name]	17-3
	PROJECT LOAD	proj_name [tech_name] [/output-name = output_name] [/readonly]	17-3
	PROJECT SAVE	[proj_name]	17-4
	PROJECT CLOSE	[/nosave]	17-5
	TECHNOLOGY SAVE	tech_pathname	17-5
INTEGER	GET_LIBRARY_CELL_COUNT	(lib_logname)	17-5
STRING	GET_LIBRARY_CELL_LIST	(lib_logname)	17-5
STRING	GET_LIBRARY_DIRNAME	(lib_logname)	17-6
STRING	GET_LIBRARY_LOGNAME	(lib_pathname)	17-7
STRING	GET_LIBRARY_PATHNAME	(lib_logname)	17-7
	CELL CREATE	cell_name [/inplace]	17-8
	CELL COPY	{cell_name} [target_lib_logname]	17-8
	CELL DELETE	{cell_name}	17-10
	CELL DISCARD	[cell_name]	17-10
	CELL EXPORT	{cell_name} file_pathname [/multifile]	17-10
	CELL IMPORT	project_pathname [{cell_name}]	17-11
	CELL OPEN	[cell_name]	17-12
	CELL NEW	cell_name	17-12
	CELL PRINT		17-13
	CELL RENAME	old_cell_name new_cell_name	17-13
	CELL SAVE	[{cell_name}]	17-14

Table 17-22:

Return type	Command/Procedure	Arguments/Named Arguments/Switches/ Parameters	Ref
	CELL SAVEAS	cell_name new_cell_name	17-14
	CELL STATUS	cell_name [/r | /rw]	17-14
	DEFINE PCELL	pcell_name [/replace] parN_name/TYPE=(type)/ DEFAULT=(value) BODY pcell_body;	17-15
BOOLEAN	CELL_EXISTS	(cell_name)	17-16
RECT	GET_CELL_BBOX	(cell_name)	17-16
INTEGER	GET_CELL_OBJECT_COUNT	(cell_name)	17-17
BOOLEAN	GET_CELL_STATUS	(cell_name, status_flag)	17-17
STRING	GET_EDITED_CELL_NAME	()	17-17
INTEGER	GET_CELL_OBJECT_COUNT	(cell_name)	17-17
	CELLVIEW REDRAW	[domain]	17-19
	REFERENCE POINT	[x] [y] [/activate | /deacti-vate]	17-19
	STATUSBAR	msg	17-20
	ZOOM	[x] [y] [x_s] [y_s] [/all | /in | /out | /selected | /window]	17-20
RECT	CELLVIEW_GET_VIEW_RECT	()	17-21
	CURRENT DIRECTORY	dir_name	17-21
	TECHNOLOGY DIRECTORY	dir_name	17-21
	EXPORT EXPERT SETTINGS	path_name	17-22
	IMPORT EXPERT SETTINGS	path_name	17-22
	RESTORE EXPERT SETTINGS	[/default | /registry]	17-22
	SAVE EXPERT SETTINGS		17-23
	SETUP GRID	/commongrid | /layergrid | / instancegrid ...	17-23
	SETUP SHAPE	[/offgridwire | /nooffgridwire] ...	17-25
	UNITS	[unit_size] [/nm | /um | /mil | / mm | /cm | /inch | /m | /lambda]	17-26
	NONSTOP MODE	[/on | /off]	17-27
	QUIET MODE	[/on | /off]	17-27

Table 17-22:

Return type	Command/Procedure	Arguments/Named Arguments/Switches/ Parameters	Ref
	VIEW MODE	[/on | /off]	17-27
	CELL ACTIVATE	[cell_name]	17-28
	CUT MODE	[/selected | /active | /all] [/ savewire | /nosavewire]	17-28
	LAYER	[layer_name]	17-28
	REGION MODE	[/normal | /merge | /hole | / holeall | /holeselected]	17-29
	SELECT MODE	[/noinstance | /instance] [/noprimitive | /primitive] [/reselect | /select | /dese- lect] [/cross | /inside] [/boundary | /solid]	17-29
	LICENCE GET	/GUARDIAN_DRC | /GUARDIAN_LPE	17-30
	LICENCE RELEASE	/GUARDIAN_DRC | /GUARDIAN_LPE	17-30
	NEW LAYER	layer_name	17-33
	DELETE LAYER	layer_name	17-33
	RENAME LAYER	old_layer_name new_layer_name	17-33
	HIDE LAYER	[layer_name]	17-34
	SHOW LAYER	[layer_name]	17-34
	SET_GDSII_NUMBER	layer_name GDSII_number	17-34
	SET_GDSII_DATATYPE	layer_name GDSII_datatype	17-35
	STYLE WIRE	[/width=] [/angle=] [/extend | / flushend | /roundend] [/extjoint | /filljoint | round- joint | /miterjoint]	17-35
	LAYERPLAN LOAD	pathname	17-35
STRING	GET_EDITED_LAYER_NAME	()	17-36
BOOLEAN	LAYER_EXISTS	(layer_name)	17-36
SEQUENCE	GET_LAYER_LIST	()	17-36
INTEGER	GET_LAYER_OBJECT_COUNT	(cell_name, layer_name)	17-37
INTEGER	GET_GDSII_NUMBER	(layer_name)	17-37
INTEGER	GET_GDSII_DATATYPE	(layer_name)	17-37
	LAYER_SET_VIEW_ATTRS	(layer_name, attr_type, flag)	17-38

Table 17-22:

Return type	Command/Procedure	Arguments/Named Arguments/Switches/ Parameters	Ref
LAYOUT-OBJECT	BOX	xpos ypos xsize ysize [layer=layername] ...	17-40
LAYOUT-OBJECT	ELLIPSE	*xcenter ycenter xradius yradius* [layer=layername] ...	17-41
LAYOUT-OBJECT	POLYGON	*{x_1,y_1,x_2,y_2,...,x_n,y_n}* [layer=layer-name] ...	17-43
LAYOUT-OBJECT	TEXT	*xpos ypos string* [layer=layername] ...	17-45
LAYOUT-OBJECT	WIRE	*{x_1,y_1,x_2,y_2,...,x_n,y_n}* [layer=layer-name] ...	17-47
LAYOUT-OBJECT	INSTANCE	xpos ypos [/cell=cellname] ...	17-49
LAYOUT-OBJECT	ARRAY	xpos ypos [/cell=cellname] [/row=value] [col=value] ...	17-51
LAYOUT-OBJECT	FLOORPLAN BLOCK	{x_1,y_1,x_2,y_2,…,x_n,y_n} name ...	17-56
LAYOUT-OBJECT	FLOORPLAN CHIP	[name=value] [/posx=] [/posy=] ...	17-54
LAYOUT-OBJECT	FLOORPLAN PAD	xpos ypos name /netname=value ...	17-56
LAYOUT-OBJECT	FLOORPLAN PIN	xpos ypos name /blockname=value /netname=value ...	17-56
	MULTIPATH	({$wire_1$,$wire_2$,…,$wire_n$})	17-57
	GROUP CREATE		17-58
	GROUP DESTROY	[/all]	17-58
	DESELECT ALL		17-61
	DESELECT OBJECT	lobject	17-62
	INVERT SELECTION		17-62
	SELECT ALL		17-62
	SELECT BOX	xpos ypos xsize ysize ...	17-62
	SELECT INSTANCES	[{cell_name}]	17-63
	SELECT LAYERS	[{layer_name}]	17-63
	SELECT NEWEST		17-63
	SELECT OBJECT	lobject	17-63

Table 17-22:

Return type	Command/Procedure	Arguments/Named Arguments/Switches/ Parameters	Ref
	COPY SELECTION		17-65
	DELETE SELECTION		17-65
	DUPLICATE SELECTION	shift_x yshift_y [repeat_x] [repeat_y]	17-65
	MIRROR SELECTION	[/x=value] [/y=value] [/duplicate]	17-66
	MOVE LAYER	[layer_name]	17-66
	MOVE SELECTION	shift_x shift_y	17-67
	ORIGIN	x y	17-67
	PASTE LAYER	[layer_name]	17-67
	PASTE SELECTION	x y	17-68
	ROTATE SELECTION	x y [angle] [/duplicate]	17-68
	REDO		17-68
	UNDO		17-70
OBJECT-CONTAINER	FIND OBJECTS	[object_type] [/criteria = {criteria}] ...	17-70
SEQUENCE	SEARCH	(object_type, criteria, search_rect, is_cross, is_selected, is_visible)	17-71
	MODIFY OBJECT	lobject [/xpos = value] [/ypos = value] [/property = {}]	17-72
	MODIFY PRIMITIVE	lobject ... [/layer=]	17-73
	MODIFY BOX	lobject ... [/xsize=] [/ysize=]	17-73
	MODIFY TEXT	lobject ... [/text=] [/height=] [/width=] [/alignstyle=] [/slope=] [/slant=] [/mirror=]	17-73
	MODIFY WIRE	lobject ... [/width=] [/endstyle=] [/jointstyle=]	17-74
	MODIFY INSTANCE	lobject [/xpos=] [/ypos=value] ... [/scale=] [/rotation=] [/mirror=] ...	17-74
	CONVERT POLYGON	lobject	17-75
	SPLIT ARRAY	lobject [/splcol] [/splrow]	17-76
	CLIP	x y x_s y_s cell_name [/out | /in]	17-76

Table 17-22:

Return type	Command/Procedure	Arguments/Named Arguments/Switches/ Parameters	Ref
	CUT LINE	{x_1,y_1, x_2,y_2,…,x_n,y_n}	17-77
	DETACH TEXT		17-77
	MERGE SELECTION		17-77
	POLYGONIZE	[/selected | /wire | /ellipse]	17-77
	RESIZE SELECTION	value [/over | /under] [/save-grid]	17-78
	SCALE	x y factor ...	17-78
	SNAP GRID	[/snap | /minimal | /visual | /custom] ...	17-79
LAYOUT-OBJECT	WIRING	[source][target][rarea=][layer=][width=][space=]	17-81
	WC_ADD_CONTACT	[cnt]/source|/target	17-82
	WC_REMOVE_CONTACTS		17-82
	EXPLODE	[/all | /selected]	17-83
	FLATTEN	project_name	17-83
	REPLACE INSTANCES	cell_name new_cell_name [/cell | /project]	17-83
	REPLACE SELECTED	cell_name	17-84
	EIP PUSH	instance [/colnum=] [/rownum=]	17-85
	EIP POP	[/save | /saveas | /explode | /reject] [/newname=] [/exit]	17-85
	DRC COMMAND	command ...	17-87
	DRC SCRIPT	pathname ...	17-89
	EXTRACT CONNECTIVITY		17-89
BOOLEAN	NODE_PICK	(pt, layer_name)	17-89
STRING	NODE_GET_NAME	()	17-90
BOOLEAN	NODE_COPY	(cell_name)	17-90
	LIBRARY ACTIVATE	lib_logname	17-90
	LIBRARY DEACTIVATE	lib_logname	17-91
	LIBRARY NAME REMOVE	lib_logname [/all]	17-91
	LIBRARY NAME SET	lib_logname lib_pathname	17-91

Table 17-22:

Return type	Command/Procedure	Arguments/Named Arguments/Switches/ Parameters	Ref
	LIBRARY TYPE	lib_pathname [/exclusive | /readonly]	17-91
	LIBRARY CHECKIN	lib_pathname	17-90
	LIBRARY CHECKOUT	lib_pathname	17-91
	LIBRARY GET VERSION	lib_pathname [/version=value]	17-92
	LIBRARY REPOSITORY ADD	repository_name lib_pathname	17-92
	LIBRARY REPOSITORY DELETE	repository_name lib_name	17-92
	LIBRARY REPOSITORY GET	repository_name lib_name pathname	17-92
SEQUENCE	GET_ACTIVATED_LIBRARY_LIST	()	17-92
STRING	GET_COMPUTER_NAME	()	17-92
SEQUENCE	GET_LIBRARY_LIST	()	17-92
SEQUENCE	GET_REPOSITORY_LIBRARY_LIST	(repository_name)	17-92
BOOLEAN	LIBRARY_ACTIVATED	(lib_logname)	17-92
	CELL CHECKOUT	[cell_name]	17-93
	CELL CHECKIN	[cell_name]	17-93
	CELL CANCEL CHECKOUT	[cell_name]	17-93
SEQUENCE	FORM_CREATE	(var_params, form_params)	17-94
BOOLEAN	FILE_EXISTS	(pathname)	17-95
STRING	FILE_SELECTOR	()	17-96
STRING	FILE_SELECTOR_EX	(params)	17-96
STRING	GET_ENVIRONMENT_STRING	(var_name)	17-98
POINT	GET_POINT	()	17-98
POINT	GET_POINT	({params})	17-98
RECT	GET_RECT	()	17-99
BOOLEAN	IS_DOUBLE	(string)	17-99
BOOLEAN	IS_FLOAT	(string)	17-100
BOOLEAN	IS_INTEGER	(string)	17-100
BOOLEAN	MESSAGE_BOX	(string, params)	17-101
	OUTSIDE_XIS	()	17-101

Table 17-22:

Return type	Command/Procedure	Arguments/Named Arguments/Switches/ Parameters	Ref
BOOLEAN	POINT_INSIDE	(pt, lobject)	17-101
INTEGER	SPAWN	program_pathname [program_arguments] [/nowait]	17-103
BOOLEAN	YES_NO_DLG	(string)	17-103
POINT	POINT_CREATE	(x, y)	17-110
RECT	RECT_CREATE	(xpos, ypos, xsize, ysize)	17-111
SEARCH-CRITE-RION	SEARCH_CRITERION_CREATE	(obj_attr, value, condition)	17-111

Chapter 18: Expert API

18.1: Overview

18.1.1: Plug-in Libraries

The EXPERT Application Programming Interface (API) provides the possibility of adding functions written in C/C++ to the EXPERT LAYOUT EDITOR. Numerous functions of API constitute very powerful and flexible means of access to full EXPERT functionality.

EXPERT API is provided in the form of a library. It contains numerous API functions that provide powerful and flexible tools to control, edit, and modify EXPERT'S project data and user interface.

Using EXPERT API, you can create one or several plug-in shared libraries. These libraries should be placed in a `plugins` directory where EXPERT'S executable module is (`ex. S:\lib\expert\4.4.4.R\x86-NT\plugins`). When starting, EXPERT loads all plug-in libraries from directory `plugins` to memory, and makes all functions from libraries available.

To access these functions, EXPERT API has to add new menu items.

The library (`CelExpApi.lib` file) is to be added into a C++ project where you build your plug-in library in the form of a Dynamically Linked Library (DLL) for Windows OS, or shared library for Unix OS. The built library is easily integrated with EXPERT'S main application.

To build a plug-in library, a C++ compiler required.

18.1.2: Structure of Objects in Expert API Library

HPROJECT is a layout project that contains an HTECHNOLOGY object and collections of HCELL objects.

HTECHNOLOGY is a technology object that contains a collection of HTLAYER objects.

HCELL is a layout cell that contains collections of HLAYER and/or HCLAYER objects.

HLAYER is a geometric layer that contains geometric HOBJECT objects (box, polygon, wire, ellipse, text) and links to HTLAYER object.

HCLAYER is a cell instance layer that contains instance HOBJECT objects (instance, array of instances, skew instance).

18.1.3: Access to Functions from Plug-in Library

For access to user functions, you have to create static objects of CCustomMenu class.

The constructor of CCustomMenu contains top level menu position, the name of the menu, and an array of `CUSTOMENU_ITEM` structures. `CUSTOMENU_ITEM` contains a description of each menu item.

It could be linked to other arrays of `CUSTOMENU_ITEM` structures, or to callback function.

Example

```
// definition of user functions
void BuildShortPath();
void DeleteShortPath();
void PrepareNetForShortSearch();
void ChangeShortDrawMode();
void SetupGraph();
void SetupPath();
```

```
// definitions of menu structures
CUSTOMENU_ITEM SetupItems[] = {
CUSTOMENU_ITEM(“Setup &Graph"    (void*)&SetupGraph),
CUSTOMENU_ITEM("Setup &Path",    (void*)&SetupPath) }

CUSTOMENU_ITEM ShortPathItems[] = {
CUSTOMENU_ITEM("&Setup", SetupItems, EA_countof(setupItems)
CUSTOMENU_ITEM("&RebuildGraph",    (void*)&TestPrepareNetForShortSearch),
CUSTOMENU_ITEM("&BuildShortPath",  (void*)&TestBuildShortPath),
CUSTOMENU_ITEM("&DeleteShortPath", (void*)&TestDeleteShortPath),
CUSTOMENU_ITEM("&ChangeDrawMode",  (void*)&TestChangeShortDrawMode) };

// static object created during loading library will insert menu items
CCustomMenu cm_ShortPath(10, "ShortPath", ShortPathItems,
EA_countof(ShortPathItems));
```

18.2: Access to Expert Settings

18.2.1: Save Settings into File

```
BOOL ExpApiExportSettings(LPCTSTR szFileName)
```

Purpose

Saves the current EXPERT settings into a text file. All settings are saved, with the exception of “Technology setup” and “Bind-key setup”, which are saved into their own files.

Parameters

`LPCTSTR szFileName`: Setting file name.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiExportSettings(“settings.ecf”);
```

18.2.2: Load Settings from File

```
BOOL ExpApiImportSettings(LPCTSTR szFileName);
```

Purpose

Loads current settings from the setting file.

Parameters

`LPCTSTR szFileName`: Setting file name.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiImportSettings(“settings.ecf”);
```

18.2.3: Set Measurement Units

```
void ExpApiSetUnits(UUNITS nUnit, double dValue = 1.0);
```

Purpose

Sets EXPERT measurement units.

Parameters

`UUNITS nUnit`	Type of units:
`UNIT_NM`	nanometers (1.0e-3 um);
`UNIT_UM`	micrometers (1.0 um);
`UNIT_MIL`	mils (2.54 um);
`UNIT_MM`	millimeters (1.0e3 um);
`UNIT_CM`	centimeters (1.0e4 um);
`UNIT_INCH`	inches (2.54e4 um);
`UNIT_M`	meters (1.0e6 um);
`UNIT_LAMBDA`	user-defined unit.
`double dValue`	value of the user-defined unit in um.

Return

None

Example

```
ExpApiSetUnits(UNIT_UM);                    // Units 1 um
ExpApiSetUnits(UNIT_LAMBDA);          // Units 1 um (by default)
ExpApiSetUnits(UNIT_LAMBDA, 0.1); // Units 100 nm (user-defined)
```

18.2.4: Get Measurement Units

```
UUNITS ExpApiGetUnits();
```

Purpose

Gets EXPERT measurement units.

Parameters

None

Return

`UUNITS`: EXPERT measurement units (see Section 18.2.3: “Set Measurement Units” function description).

Example

```
UUNITS uNumber = ExpApiGetUnits();
```

18.2.5: Get Measurement Unit Value in Microns

```
double ExpApiGetUnitUMValue(UUNITS nUnit = UNIT_NONE);
```

Purpose

Returns value in `um` of the unit specified by the argument. If no argument is given, or the argument is `UNIT_NONE`, the micron value of the current EXPERT unit is returned.

Parameters

`UUNITS nUnit`: Type of unit (see Section 18.2.3: "Set Measurement Units" function description).

Return

`double`: value in `um` for the requested unit.

Example

```
double dNum = ExpApiGetUnitUMValue(UNIT_NM); // value in um of UNIT_NM
double dNum = ExpApiGetUnitUMValue(); // value in um of the current Expert
unit
```

18.2.6: Set Drawing Angle Mode

```
BOOL ExpApiSetAngleMode(DAM daMode);
```

Purpose

Sets Drawing Angle Mode.

Parameters

`DAM daMode`: New Drawing Mode.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetAngleMode(DAM_90D);
```

18.2.7: Get Angle Mode

```
DAM ExpApiGetAngleMode();
```

Purpose

Retrieves the current Drawing Angle Mode.

Parameters

None

Return

`DAM`: current Drawing Mode.

Example

```
DAM daMode = ExpApiSetAngleMode();
```

18.2.8: Set Region Mode

```
BOOL ExpApiSetRegionMode(int nRegionMode);
```

Purpose

Sets Region Mode.

Parameters

`int nRegionMode`: A Region Mode to be set.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetRegionMode(RM_ HOLE);
```

18.2.9: Get Region Mode

```
int ExpApiGetRegionMode();
```

Purpose

Retrieves current Region Mode.

Parameters

None

Return

`int`: current Region Mode.

Example

```
int nRegionMode = ExpApiGetRegionMode();
```

18.2.10: Set General Setup Data

```
void ExpApiSetSetupGeneral(const GENERAL_SETUP& sGeneralSetup);
```

Purpose

Sets EXPERT'S General Setup data.

Parameters

`GENERAL_SETUP& sGeneralSetup`: Reference to a General Setup structure.

Return

None

Example

```
GENERAL_SETUP sGeneralSetup;
ExpApiGetSetupGeneral(sGeneralSetup);
sGeneralSetup.someField = … some value … ;
ExpApiSetSetupGeneral(sGeneralSetup);
```

18.2.11: Get General Setup Data

```
void ExpApiGetSetupGeneral(GENERAL_SETUP& sGeneralSetup);
```

Purpose

Gets EXPERT'S General Setup data.

Parameters

`GENERAL_SETUP& sGeneralSetup`: Reference to a General Setup structure.

Return

None

Example

```
GENERAL_SETUP sGeneralSetup;
ExpApiGetSetupGeneral(sGeneralSetup);
```

18.2.12: Set Technology Setup Data

```
void ExpApiSetSetupTechnology (const TECHNOLOGY_SETUP& sTechSetup);
```

Purpose

Sets EXPERT'S Technology Setup data.

Parameters

`TECHNOLOGY_SETUP & sTechSetup`: Reference to a Technology Setup structure.

Return

None

Example

```
TECHNOLOGY_SETUP sTechSetup;
ExpApiGetSetupTechnology (sTechSetup);
sTechSetup.someField = … some value … ;
ExpApiSetSetupTechnology (sTechSetup);
```

18.2.13: Get Technology Setup Data

```
void ExpApiGetSetupTechnology (TECHNOLOGY_SETUP& sTechSetup);
```

Purpose

Gets EXPERT'S Technology Setup data.

Parameters

`TECHNOLOGY_SETUP & sTechSetup`: Reference to a Technology Setup structure.

Return

None

Example

```
TECHNOLOGY_SETUP sTechSetup;
ExpApiGetSetupTechnology (sTechSetup);
```

18.2.14: Set GDSII Setup Data

```
void ExpApiSetSetupGDSII(const GDSII_SETUP& sGDSIISetup);
```

Purpose

Sets EXPERT'S GDSII Setup data.

Parameters

`GDSII_SETUP& sGDSIISetup`: Reference to a GDSII Setup structure.

Return

None

Example

```
GDSII_SETUP sGDSIISetup;
sGDSIISetup.someField = … some data … ;
ExpApiSetSetupGDSII (sGDSIISetup);
```

18.2.15: Get GDSII Setup Data

```
void ExpApiGetSetupGDSII(GDSII_SETUP& sGDSIISetup);
```

Purpose

Gets EXPERT'S GDSII Read/Write Setup data.

Parameters

`GDSII_SETUP& sGDSIISetup`: Reference to a GDSII Setup structure.

Return

None

Example

```
GDSII_SETUP sGDSIISetup;
ExpApiGetSetupGDSII (sGDSIISetup);
```

18.2.16: Set CIF Setup Data

```
void ExpApiSetSetupCIF(const CIF_SETUP& sCIFSetup);
```

Purpose

Sets EXPERT'S CIF Read/Write Setup data.

Parameters

`CIF_SETUP& sCIFSetup`: Reference to a CIF Setup structure.

Return

None

Example

```
CIF_SETUP sCIFSetup;
sCIFSetup.someField  = … some data … ;
ExpApiSetSetupCIF(CIFSetup);
```

18.2.17: Get CIF Setup Data

```
void ExpApiGetSetupCIF(CIF_SETUP& sCIFSetup);
```

Purpose

Gets EXPERT'S CIF Setup data.

Parameters

`CIF_SETUP& sCIFSetup`: Reference to a CIF Setup structure.

Return

None

Example

```
CIF_SETUP sCIFSetup;
ExpApiGetSetupCIF(CIFSetup);
```

18.2.18: Set Applicon Setup Data

```
void ExpApiSetSetupApplicon(const APPLICON_SETUP& sAPPLICONSetup);
```

Purpose

Sets Expert's Applicon Read/Write Setup data.

Parameters

`APPLICON_SETUP& sAPPLICONSetup`: Reference to an Applicon Setup structure.

Return

None

Example

```
APPLICON_SETUP sAPPLICONSetup;
sAPPLICONSetup.someField  = … some data … ;
ExpApiSetSetupApplicon(sAPPLICONSetup);
```

18.2.19: Get Applicon Setup Data

```
void ExpApiGetSetupApplicon(APPLICON_SETUP& sAPPLICONSetup);
```

Purpose

Gets EXPERT'S Applicon Setup data.

Parameters

`APPLICON_SETUP& sAPPLICONSetup`: Reference to an Applicon Setup structure.

Return

None

Example

```
APPLICON_SETUP sAPPLICONSetup;
ExpApiGetSetupApplicon(sAPPLICONSetup);
```

18.3: Project

18.3.1: Is Project Open

```
BOOL ExpApiIsProjectOpen();
```

Purpose

Checks if there is a project open in EXPERT.

Parameters

None

Return

BOOL: TRUE, if a project is open; FALSE otherwise.

Example

```
BOOL bResult = ExpApiIsProjectOpen();
```

18.3.2: New Project

```
BOOL ExpApiProjectNew(LPCTSTR szProjName, LPCTSTR szTechName = NULL);
```

Purpose

Creates a new Project with the specified Technology or empty technology, if Technology name is omitted or NULL.

Parameters

LPCTSTR szProjName: Project name;

LPCTSTR szTechName: Technology file name.

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiProjectNew("New_Project.eld", "New_Technology.tcn");
```

18.3.3: Load Project

```
BOOL ExpApiProjectLoad(LPCTSTR szProjName, LPCTSTR szOutputProjectName =
NULL, LPCTSTR szTechName = NULL, BOOL bReadOnly = FALSE);
```

Purpose

Loads a Project with specified Technology, if the Technology name is not NULL, and saves Output/ Backup file, if not NULL.

Parameters

LPCTSTR szProjName: Input file name.

The format of the file is determined by its extension. Lists of allowable input file extensions are specified in the corresponding setups (APPLICON_SETUP, GDSII_SETUP, CIF_SETUP).

LPCTSTR szOutputProjectName: Output Project name for *.GDS, *.CIF or *.APL types of loading file; backup file name for *ELD.

LPCTSTR szTechName: Technology file name.

BOOL bReadOnly: Read-only flag.

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiProjectLoad(“Proj_1.eld”, “Proj_1a.eld”, “Techno.tcn”,
```

```
FALSE);
```

18.3.4: Close Project

```
BOOL ExpApiProjectClose ();
```

Purpose

Closes the current Project.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiProjectClose ();
```

18.3.5: Save Project

```
BOOL ExpApiProjectSave();
```

Purpose

Saves the current Project, if it is changed.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiProjectSave();
```

18.3.6: Save Project As Another File

```
BOOL ExpApiProjectSaveAs(LPCTSTR szProjName);
```

Purpose

Saves current Project into a file with the specified name.

The format of the file is determined by its extension. Lists of allowable output file extensions are specified in the corresponding setups (`APPLICON_SETUP`, `GDSII_SETUP`, `CIF_SETUP`).

Parameters

`LPCTSTR szProjName`: File name.

Return

BOOL - TRUE if successful; FALSE otherwise.

Example:

```
BOOL bResult = ExpApiProjectSaveAs ("New_Project.eld");
```

18.3.7: Compress Project

```
BOOL ExpApiProjectCompress ();
```

Purpose

Decreases the size of the current project data file by performing “garbage collection” operation.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiProjectCompress ();
```

18.4: Technology

18.4.1: Get the Number of Technology Layers

```
int ExpApiGetTechLayerNumber();
```

Purpose

Retrieves the number of Technology Layers.

Parameters

None

Return

`int`: the number of Technology Layers.

Example

```
int nNum = ExpApiGetTechLayerNumber();
```

18.4.2: Get First Technology Layer

```
HTLAYER ExpApiGetTechFirstLayer();
```

Purpose

Returns handle to the first Technology Layer.

Parameters

None

Return

`HTLAYER`: The first Technology Layer.

Example

```
HTLAYER hLayer = ExpApiGetTechFirstLayer();
```

18.4.3: Get Next Technology Layer

```
HTLAYER ExpApiGetTechNextLayer();
```

Purpose

Returns the handle to the next Technology Layer or `EA_INVALID_HANDLE`, if no layers left.

Parameters

None

Return

`HTLAYER`: The next Technology Layer.

Example

```
HTLAYER hLayer = ExpApiGetTechNextLayer();
```

18.4.4: Get Technology Layer Name

```
LPCTSTR ExpApiGetTechLayerName(HTLAYER hLayer);
```

Purpose

Returns the name of a Technology Layer given by its handle.

Parameters

`HTLAYER hLayer`: Technology Layer handle.

Return

LPCTSTR: The name of the Technology Layer.

Example

```
HTLAYER hLayer = … some data …;
LPCTSTR szLayer = ExpApiGetTechLayerName(hLayer);
```

18.4.5: Set Technology Layer

```
BOOL ExpApiSetTechLayer(LPCTSTR szLayerName, const LAYER_PROPERTY&
Property);
```

Purpose

Sets the Property (specification) of the Technology Layer specified by its name.

Parameters

`LPCTSTR szLayerName`: Layer name;

`LAYER_PROPERTY& Property`: Layer property.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
LAYER_PROPERTY Property;
Property.someField  = … some data …;
LPCTSTR szName = "Layer_1";
BOOL bResult = ExpApiSetTechLayer(szName, Property);
```

18.4.6: Get Technology Layer Specification

```
BOOL ExpApiGetTechLayer(LPCTSTR szLayerName, LAYER_PROPERTY& Property);
```

Purpose

Gets the Property of specified Technology Layer.

Parameters

`LPCTSTR szLayerName`: Layer name;

`LAYER_PROPERTY& Property`: Layer property.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
LAYER_PROPERTY Property;
LPCTSTR szName = "Layer_1";
BOOL bResult = ExpApiGetTechLayer(szName, Property);
```

18.4.7: Add Technology Layer

```
BOOL ExpApiAddTechLayer(const LAYER_PROPERTY& Property);
```

Purpose

Creates a new Technology Layer with specified Property.

Parameters

`LAYER_PROPERTY& Property`: Technology Layer property.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
LAYER_PROPERTY Property;
Property.someField  = … some data …;
BOOL bResult = ExpApiAddTechLayer(Property);
```

18.4.8: Delete Technology Layer

```
BOOL ExpApiDeleteTechLayer(LPCTSTR szLayerName);
```

Purpose

Deletes specified Technology Layer.

Parameters

`LPCTSTR szLayerName`: Technology Layer name.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
LPCTSTR szName = “Layer_1”;
BOOL bResult = ExpDeleteTechLayer(szName);
```

18.4.9: Get Technology Layer Handle

```
HTLAYER ExpApiGetTechLayerName(LPCTSTR szLayer);
```

Purpose

Returns the handle of a Technology Layer given by its name, or `EA_INVALID_HANDLE` if this layer doesn't exist.

Parameters

`LPCTSTR`: The name of the Technology Layer.

Return

`HTLAYER hLayer`: Technology Layer handle.

Example

```
LPCTSTR szLayer = ... some data ...;
HTLAYER hLayer = ExpApiGetTechLayerHandle(szLayer);
```

18.5: Cells

18.5.1: Set Current Cell

```
BOOL ExpApiSetCurrentCell(LPCTSTR szCellName);
```

Purpose

Makes the specified cell active.

Parameters

`LPCTSTR szCellName`: Name of the cell.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetCurrentCell( "Cell_1");
```

18.5.2: Get Current Cell

```
LPCTSTR ExpApiGetCurrentCell();
```

Purpose

Retrieves the name of the active cell.

Parameters

None

Return

LPCTSTR: The name of the active cell.

Example

```
LPCTSTR szName = ExpApiGetCurrentCell();
```

18.5.3: Get Cell Number

```
int ExpApiGetCellNumber(LPCTSTR szLibName = NULL);
```

Purpose

Gets the number of cells in the specified Library or current Project, if Library name is omitted or NULL.

Parameters

`LPCTSTR szLibName`: Logical library name or NULL.

Return

`int`: Total number of cells.

Example

```
int nNum = ExpApiGetCellNumber();
```

18.5.4: Get First Cell

```
HCELL ExpApiGetFirstCell(LPCTSTR szLibName = NULL);
```

Purpose

Returns handle to the first cell in the specified Library or current Project, if Library name is omitted or NULL.

Parameters

`LPCTSTR szLibName`: Logical library name.

Return

`HCELL`: Handle to the first cell.

Example

```
HCELL hCell = ExpApiGetFirstCell();
```

18.5.5: Get Next Cell

```
HCELL ExpApiGetNextCell(LPCTSTR szLibName = NULL);
```

Purpose

Returns handle to the next cell in the specified Library or current Project, if Library name is omitted or NULL.

Parameters

`LPCTSTR szLibName`: Logical library name.

Return

`HCELL`: Handle to the next cell or `EA_INVALID_HANDLE`, if no cells left.

Example

```
HCELL hCell = ExpApiGetNextCell();
```

18.5.6: Get Cell Name

```
LPCTSTR ExpApiGetCellName(HCELL hCell);
```

Purpose

Retrieves cell's name by cell's handle.

Parameters

`HCELL hCell`: Cell handle.

Return

`LPCTSTR`: Name of the cell.

Example

```
HCELL hCell = … some data … ;
LPCTSTR szCellName = ExpApiGetCellName(hCell);
```

18.5.7: Create New Cell

```
BOOL ExpApiCreateNewCell(LPCTSTR szCellName);
```

Purpose

Creates a new cell with the specified name.

Parameters

`LPCTSTR szCellName`: The name of the new cell to create.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiCreateNewCell("New_Cell");
```

18.5.8: Create New Cell from Rectangular Clipping

```
BOOL ExpApiCreateNewClipCell (LPCTSTR szCellName, RCEARect ptRect, BOOL
bClipIn = TRUE);
```

Purpose

Creates a new cell from all layout geometry in the specified Rectangle with maximally preserved cell hierarchy.

Parameters

`LPCTSTR szCellName`: Cell name;

`RCEARect ptRect`: Rectangle to ClipIn/ClipOut;

`BOOL bClipIn`: TRUE, if to make Clip In (replace the piece of layout by the instance of the new cell); FALSE to make Clip Out (clip out a piece of layout to a new cell).

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
CEARect ptRect = CEARect(CEAPoint(1000.0, 1000.0), CEAPoint(2000.0,
2000.0));
BOOL bResult = ExpApiCreateNewClipCell ("Cell_ClipOut", ptRect, FALSE);
```

18.5.9: Delete Cell

```
BOOL ExpApiDeleteCell(LPCTSTR szCellName = NULL, BOOL bHier = FALSE);
```

Purpose

Deletes the cell with the specified name or the current cell, if cell name is omitted, along with or without its hierarchy.

Parameters

LPCTSTR szCellName: The name of the cell to delete;

BOOL bHier -TRUE: Delete cell together with all subcells that will no longer be in use.

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiDeleteCell(“Cell_1”, FALSE);
```

18.5.10: Rename Cell

```
BOOL ExpApiRenameCell(LPCTSTR szTargetCellName, LPCTSTR szCellName = NULL);
```

Purpose

Renames the cell with the specified name or the current cell, if cell name is omitted, to the Target cell name.

Parameters

LPCTSTR szTargetCellName: New cell name;

LPCTSTR szCellName: Name of cell to rename.

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiRenameCell(“New_Cell”, “Cell”);
```

18.5.11: Save Cell

```
BOOL ExpApiSaveCell(LPCTSTR szCellName = NULL);
```

Purpose

Saves cell with specified name or current cell, if cell name is omitted.

Parameters

`LPCTSTR szCellName`: Name of cell to save.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSaveCell("Cell_1")
```

18.5.12: Save As Cell

```
BOOL ExpApiSaveAsCell(LPCTSTR szTargetCellName, LPCTSTR szCellName = NULL);
```

Purpose

Saves the contents of the cell with the specified name or the current cell, if cell name is omitted, into a new cell with the Target name.

Parameters

`LPCTSTR szTargetCellName`: Cell name to save contents to;

`LPCTSTR szCellName`: Cell name to save contents from.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSaveAsCell("New_Cell", "Old_Cell");
```

18.5.13: Save Selected As Cell

```
BOOL ExpApiSaveSelectedAsCell(LPCTSTR szCellName, BOOL bReplace = TRUE);
```

Purpose

Saves selected Objects into a new cell with the specified name.

Parameters

`LPCTSTR szCellName`: Cell name.

`BOOL bReplace`: TRUE, if to replace the selected objects by the instance of the new cell.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSaveSelectedAsCell("Cell_ClipIn");
BOOL bResult = ExpApiSaveSelectedAsCell("Cell_ClipOut", FALSE);
```

18.5.14: Discard Cell

```
BOOL ExpApiDiscardCell(LPCTSTR szCellName = NULL);
```

Purpose

Discards recent changes in the specified cell or current cell, if cell name is omitted.

Parameters

`LPCTSTR szCellName`: Name of the cell to discard changes.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiDiscardCell(“Cell_1”);
```

18.5.15: Set Cell Read-only Flag

```
BOOL ExpApiSetReadOnlyCell(BOOL bReadOnly = TRUE, LPCTSTR szCellName = NULL);
```

Purpose

Sets Read-only flag for specified cell or current cell, if cell name is omitted.

Parameters

`BOOL bReadOnly`: TRUE if Read-only; FALSE if write-enabled;

`LPCTSTR szCellName`: Cell name.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetReadOnlyCell(TRUE, “Cell_1”);
```

18.5.16: Get Cell Read-only Flag

```
BOOL    ExpApiGetReadOnlyCell(LPCTSTR szCellName = NULL);
```

Purpose

Gets value of Read-only flag for specified cell or current cell, if cell name is omitted.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`BOOL`: TRUE if cell is Read-only; FALSE if write-enabled.

Example

```
BOOL bRO = ExpApiGetReadOnlyCell(“Cell_1”);
```

18.5.17: Set Origin

```
BOOL ExpApiSetOrigin(double dx, double dy, LPCTSTR szCellName = NULL);
BOOL ExpApiSetOrigin(RCEAPoint pt, LPCTSTR szCellName = NULL);
```

Purpose

Sets new Origin of specified cell or current cell, if cell name is omitted.

Parameters

`double dx, double dy`: Coordinates of new Origin;

`RCEAPoint pt`: Coordinates of new Origin;

`LPCTSTR szCellName`: Cell name.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetOrigin(100, 200); // for the current cell
BOOL bResult = ExpApiSetOrigin(CEAPoint(100, 200), "Cell_1");
```

18.5.18: Set Reference Point

```
void ExpApiSetRefPoint(RCEAPoint ptStart, LPCTSTR szCellName = NULL);
```

Purpose

Sets new Reference Point coordinates for specified cell or current cell, if cell name is omitted.

Parameters

`RCEAPoint ptStart`: Position of the Reference Point;

`LPCTSTR szCellName`: Cell name.

Return

None

Example

```
ExpApiSetRefPoint(CEAPoint(100, 200));
ExpApiSetRefPoint(CEAPoint(100, 200), "Cell_1");
```

18.5.19: Get Reference Point

```
CEAPoint ExpApiGetRefPoint(LPCTSTR szCellName = NULL);
```

Purpose

Gets current Reference Point coordinates for specified cell or current cell, if cell name is omitted.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`CEAPoint`: Position of the Reference Point.

Example

```
CEAPoint pt = ExpApiGetRefPoint();
CEAPoint pt = ExpApiGetRefPoint("Cell_1");
```

18.5.20: Activate Reference Point

```
BOOL ExpApiActivateRefPoint(BOOL bRefPointActivate = TRUE, LPCTSTR
szCellName = NULL);
```

Purpose

Activates/deactivates the Reference Point for specified cell or current cell, if cell name is omitted.

Parameters

`BOOL bRefPointActivate`: Flag (TRUE to activate reference point; FALSE to deactivate);

`LPCTSTR szCellName`: Cell name.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiActivateRefPoint(); // Activate reference point
BOOL bResult = ExpApiActivateRefPoint(FALSE, “Cell_1”); // Deactivate
reference point
```

18.5.21: Get Window Rect

```
CEARect ExpApiGetCurrentWindowRect();
```

Purpose

Gets current window rectangle, i.e. drawing area size, in user units.

Parameters

None

Return

`CEARect`: Drawing area.

Example

```
CEARect r = ExpApiGetCurrentWindowRect();
```

18.5.22: Is Ancestor of Cell

```
BOOL ExpApiIsAncestorOf(LPCTSTR szCellName, LPCTSTR szParentCellName);
```

Purpose

Answers if Cell is ancestor of ParentCell.

Parameters

`LPCTSTR szCellName`: Prospective ancestor cell;

`LPCTSTR szParentCellName`: Parent cell name.

Return

`BOOL`: TRUE if ancestor; FALSE otherwise.

Example

```
BOOL bResult = ExpApiIsAncestorOf (“Cell”, “CellQ”);
```

18.6: Layers

18.6.1: Set Current Layer

```
BOOL ExpApiSetCurrentLayer(LPCTSTR szLayerName);
```

Purpose

Makes specified Layer active.

Parameters

`LPCTSTR szLayerName`: Name of the layer.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetCurrentLayer("Layer_1");
```

18.6.2: Get Current Layer

```
LPCTSTR ExpApiGetCurrentLayer();
```

Purpose

Retrieves the name of the active Layer.

Parameters

None

Return

`LPCTSTR`: Name of active Layer.

Example

```
LPCTSTR szName = ExpApiGetCurrentLayer();
```

18.6.3: Get Layer Number

```
int ExpApiGetLayerNumber(LPCTSTR szCellName = NULL);
```

Purpose

Gets the number of Layers that contain any data in the specified cell or the current cell, if cell name is omitted.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`int`: Number of Layers.

Example

```
int nLayers = ExpApiGetLayerNumber();
```

18.6.4: Get First Layer

```
HLAYER ExpApiGetFirstLayer(LPCTSTR szCellName = NULL);
```

Purpose

Returns handle to the first Layer with data in specified cell or current cell, if cell name is omitted.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`HLAYER`: Handle to the first data Layer in the cell.

Example

```
HLAYER hLayer = ExpApiGetFirstLayer("Cell_1");
```

18.6.5: Get Next Layer

```
HLAYER ExpApiGetNextLayer(LPCTSTR szCellName = NULL);
```

Purpose

Returns handle to the next Layer in specified cell or current cell, if cell name is omitted.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`HLAYER`: Handle to the next data Layer in the cell or EA_INVALID_HANDLE, if nothing left.

Example

```
HLAYER hLayer = ExpApiGetNextLayer("Cell_1");
```

18.6.6: Get Layer Name

```
LPCTSTR ExpApiGetLayerName(HLAYER hLayer);
```

Purpose

Returns Layer's name by Layer's handle.

Parameters

`HLAYER hLayer`: Layer handle.

Return

`LPCTSTR`: The name of Layer

Example

```
HLAYER hLayer = … some data …;
LPCTSTR szLayer = ExpApiGetLayerName(hLayer);
```

18.6.7: Get Cell Layer Number

```
int ExpApiGetCellLayerNumber(LPCTSTR szCellName = NULL);
```

Purpose

Retrieves number of cell instance/array layers in the specified cell or current cell, if cell name is omitted.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`int`: Number of Cell Layers in the cell.

Example

```
int nNum = ExpApiGetCellLayerNumber();
```

18.6.8: Get Cell First Layer

```
HCLAYER ExpApiGetFirstCellLayer (LPCTSTR szCellName = NULL);
```

Purpose

Returns handle of the first Cell Layer in specified cell or current cell, if cell name is omitted.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`HCLAYER`: Handle of the first Cell Layer.

Example

```
HCLAYER hLayer = ExpApiGetFirstCellLayer ("Cell_1");
```

18.6.9: Get Cell Next Layer

```
HCLAYER ExpApiGetNextCellLayer (LPCTSTR szCellName = NULL);
```

Purpose

Returns handle of the next Cell Layer in specified cell or current cell, if cell name is omitted.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`HCLAYER`: Handle of the next Cell Layer or EA_INVALID_HANDLE if no cell layers left.

Example

```
HCLAYER hLayer = ExpApiGetNextCellLayer ("Cell_1");
```

18.6.10: Get Cell Layer Name

```
LPCTSTR ExpApiGetCellLayerName(HCLAYER hLayer);
```

Purpose

Returns Cell Layer's name by Layer's handle.

Parameters

`HCLAYER hLayer`: Layer handle.

Return

`LPCTSTR`: The name of Cell Layer.

Example

```
HCLAYER hLayer = … some data …;
LPCTSTR szLayer = ExpApiGetCellLayerName(hLayer);
```

18.6.11: Generate Derived Layer

```
BOOL ExpApiGenDLayer(LPCTSTR szLayerName = NULL);
```

Purpose

Generates Derived Layer with specified name or the current layer, if Derived Layer name is omitted.

Parameters

`LPCTSTR szLayerName`: Name of derived layer.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiGenDLayer("New_DLayer");
```

18.6.12: Generate All Derived Layers

```
BOOL ExpApiGenAllDLayers();
```

Purpose

Generates all Derived Layers.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = BOOL ExpApiGenAllDLayers();
```

18.6.13: Clean Cell Primitives in Derived Layers

```
BOOL ExpApiCleanCellPrimitivesDLayers();
```

Purpose

Deletes data from all derived layers of the current cell.

Parameters

None

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiCleanCellPrimitivesDLayers();
```

18.6.14: Clean Cell Hierarchy in Derived Layers

```
BOOL ExpApiCleanCellHierarchyDLayers();
```

Purpose

Deletes data from all derived layers of the current cell and of all its ancestor cells.

Parameters

None

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiCleanCellHierarchyDLayers();
```

18.6.15: Clean Derived Layers in the Whole Project

```
BOOL ExpApiCleanWholeProjectDLayers();
```

Purpose

Cleans Derived Layers in the whole project.

Parameters

None

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiCleanWholeProjectDLayers();
```

18.6.16: Clean Cell Primitives in Scratch Layers

```
BOOL ExpApiCleanCellPrimitivesSLayers();
```

Purpose

Deletes data from all scratch layers of the current cell.

Parameters

None

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiCleanCellPrimitivesSLayers();
```

18.6.17: Clean Cell Hierarchy in Scratch Layers

```
BOOL ExpApiCleanCellHierarchySLayers();
```

Purpose

Deletes data from all scratch layers of the current cell and of all its ancestor cells.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiCleanCellHierarchySLayers();
```

18.6.18: Clean the Whole Project in Scratch Layers

```
BOOL ExpApiCleanWholeProjectSLayers()
```

Purpose

Cleans Scratch Layers in the whole Project.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiCleanWholeProjectSLayers();
```

18.6.19: Delete Scratch Layers in Technology

```
BOOL ExpApiDeleteTechnologySLayers();
```

Purpose

Deletes Scratch Layers from Technology.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiDeleteTechnologySLayers();
```

18.7: Layout Objects

18.7.1: Get Object Specs

```
void ExpApiGetObjectProperty(HOBJECT hObject, OBJECT_PROPERTY& Property);
```

Purpose

Gets Property of the specified Object.

Parameters

`HOBJECT hObject`: Handle to Object;

`OBJECT_PROPERTY& Property`: Reference to Object Property.

Return

None

Example

```
HOBJECT hObject = … some object … ;
OBJECT_PROPERTY Property;
ExpApiGetObjectProperty(hObject, Property);
```

18.7.2: Add Object

```
HOBJECT ExpApiAddObject(OBJECT_PROPERTY& Property);
```

Purpose

Adds a new Object with specified property.

Parameters

`OBJECT_PROPERTY& Property`: Reference to Object property.

Return

`HOBJECT hObject`: Handle to added Object.

Example

```
OBJECT_PROPERTY Property;
Property.someField  = … some data … ;
HOBJECT hObject = ExpApiAddObject(Property);
```

18.7.3: Delete Object

```
BOOL ExpApiDeleteObject(HOBJECT hObject);
```

Purpose

Deletes the specified Object.

Parameters

`HOBJECT hObject`: Handle to Object to delete.

Return

`BOOL`: TRUE if pressed; FALSE otherwise.

Example

```
HOBJECT hObject = … some object … ;
BOOL bResult = ExpApiDeleteObject(hObject);
```

18.7.4: Modify Object

```
HOBJECT ExpApiModifyObject(HOBJECT hObject, OBJECT_PROPERTY& Property);
```

Purpose

Modifies Object's property.

Parameters

`HOBJECT hObject`: Handle to Object to modify;

`OBJECT_PROPERTY& Property`: Reference to Object property.

Return

`HOBJECT hObject`: Handle to modified Object.

Example

```
HOBJECT hObject = … some object … ;
OBJECT_PROPERTY Property;
Property.someField = … some data … ;
ExpApiModifyObject(hObject, Property);
```

18.7.5: Search

```
int ExpApiSearch(const CEASearchContext& SearchContext, LPCTSTR szCellName =
NULL);
```

Purpose

Searches objects in specified cell or current cell, if cell name is omitted, for specific context (search criteria).

For access to found objects, use `ExpApiGetFoundFirstObject()`, `ExpApiGetFoundNextObject()`, `ExpApiGetFoundPrevObject()`, `ExpApiGetFoundLastObject()` functions.

Parameters

`const CEASearchContext& SearchContext`: Search Context.

`LPCTSTR szCellName`: Cell name to search for Context.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
CEASearchContext SearchContext;
SearchContext.m_nTypeSearchObject = SOT_ELLIPSE;
int nFound = ExpApiSearch(SearchContext, "Cell_1");
```

18.7.6: Find First Object

```
HOBJECT ExpApiGetFoundFirstObject(LPCTSTR szCellName = NULL);
```

Purpose

Gets first Object in specified cell or current cell, if cell name is omitted. The `ExpApiSearch()` call is required before calling this function.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`HOBJECT hObject`: Handle to found first Object.

Example

```
HOBJECT hObject = ExpApiGetFoundFirstObject("Cell_1");
```

18.7.7: Find Next Object

```
HOBJECT ExpApiGetFoundNextObject(LPCTSTR szCellName = NULL);
```

Purpose

Gets next Object in specified cell or current cell, if cell name is omitted.

The `ExpApiSearch()` and either `ExpApiGetFoundFirstObject()`, or `ExpApiGetFoundLastObject()` calls are required before calling this function.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`HOBJECT hObject`: Handle to found next Object or EA_INVALID_OBJECT.

Example

```
HOBJECT hObject = ExpApiGetFoundNextObject("Cell_1");
```

18.7.8: Find Previous Object

```
HOBJECT ExpApiGetFoundPrevObject(LPCTSTR szCellName = NULL);
```

Purpose

Gets previous Object in specified cell or current cell, if cell name is omitted. The `ExpApiSearch()` and either `ExpApiGetFoundFirstObject()`, or `ExpApiGetFoundLastObject()` calls are required before calling this function.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`HOBJECT hObject`: Handle to previous found Object or `EA_INVALID_OBJECT`.

Example

```
HOBJECT hObject = ExpApiGetFoundPrevObject("Cell_1");
```

18.7.9: Find Last Object

```
HOBJECT ExpApiGetFoundLastObject(LPCTSTR szCellName = NULL);
```

Purpose

Gets last Object in specified cell or current cell, if cell name is omitted. The `ExpApiSearch()` call is required before calling this function.

Parameters

`LPCTSTR szCellName`: Cell name.

Return

`HOBJECT hObject`: Handle to last found Object.

Example

```
HOBJECT hObject = ExpApiGetFoundLastObject("Cell_1");
```

18.7.10: Get First Object

```
HOBJECT ExpApiGetFirstObject(LPCTSTR szLayerName = NULL, LPCTSTR szCellName
= NULL);
```

Purpose

Gets first Object in specified Layer or current Layer, if Layer name is omitted, and specified cell or current cell, if cell name is omitted.

Parameters

`LPCTSTR szLayerName`: Layer name;

`LPCTSTR szCellName`: Cell name.

Return

`HOBJECT hObject`: Handle of the first Object.

Example

```
HOBJECT hObject = ExpApiGetFirstObject("Layer_1", "Cell_1");
```

18.7.11: Get Next Object

```
HOBJECT ExpApiGetNextObject(LPCTSTR szLayerName = NULL, LPCTSTR szCellName =
NULL);
```

Purpose

Gets next Object in specified Layer or current Layer, if Layer name is omitted, and specified cell or current cell, if cell name is omitted. The `ExpApiGetFirstObject()` call is required before calling this function.

Parameters

`LPCTSTR szLayerName`: Layer name;

`LPCTSTR szCellName`: Cell name.

Return

`HOBJECT hObject`: Handle of the next Object or `EA_INVALID_OBJECT`.

Example

```
HOBJECT hObject = ExpApiGetNextObject("Layer_1", "Cell_1");
```

18.7.12: Get First Cell Instance/Array Object

```
HOBJECT ExpApiGetFirstInstObject(LPCTSTR szInstName = NULL, LPCTSTR
szCellName = NULL);
```

Purpose

Gets first Instance/array of the cell with name szInstName or of any cell, if `szInstName` is NULL or omitted, in the specified cell or current cell, if `szCellName` is NULL or omitted.

Parameters

`LPCTSTR szInstName`: Instance name;

`LPCTSTR szCellName`: Cell name.

Return

`HOBJECT hObject`: Handle to the first instance of Object.

Example

```
HOBJECT hObject = ExpApiGetFirstInstObject("Instance_1", "Cell_1");
```

18.7.13: Get Next Cell Instance/Array Object

```
HOBJECT ExpApiGetNextInstObject(LPCTSTR szInstName = NULL, LPCTSTR
szCellName = NULL);
```

Purpose

Gets next Instance/array of the cell with name szInstName or of any cell, if `szInstName` is NULL or omitted, in the specified cell or current cell, if `szCellName` is NULL or omitted.

`ExpApiGetFirstInstObject()` call is required before calling this function.

Parameters

`LPCTSTR szInstName`: Instance name;

`LPCTSTR szCellName`: Cell name.

Return

`HOBJECT hObject`: Handle to the next instance of Object or `EA_INVALID_OBJECT`.

Example

```
HOBJECT hObject = ExpApiGetNextInstObject("Instance_1", "Cell_1");
```

18.7.14: Box

```
HOBJECT ExpApiCreateBox(double dSx, double dSy, double dEx, double dEy,
LPCTSTR szLayerName = NULL);
HOBJECT ExpApiCreateBox(RCEAPoint ptPos, RCEAPoint ptSize, LPCTSTR
szLayerName = NULL);
HOBJECT ExpApiCreateBox(RCEARect rtRect, LPCTSTR szLayerName = NULL);
```

Purpose

Creates a Box in specified Layer or current Layer, if Layer name is omitted.

Parameters

`double dSx`, `double dSy`: Initial position of Bottom Left corner;

`RCEAPoint ptPos`: Initial position of Bottom Left corner;

`double dEx`, `double dEy`: width, height;

`RCEAPoint ptSize`: Width, height;

`RCEARect rtRect`: Initial position of Bottom Left corner, width, height;

`LPCTSTR szLayerName`: name of the Layer.

Return

`HOBJECT`: Handle to the created box; `EA_INVALID_OBJECT` if failed.

Example

```
HOBJECT hObject = ExpApiCreateBox(0, 0, 100, 100, "Layer_1");
```

18.7.15: Circle

```
HOBJECT ExpApiCreateCircle(double dSx, double dSy, double dRadius, LPCTSTR
szLayerName = NULL);
HOBJECT ExpApiCreateCircle(RCEAPoint ptCenter, double dRadius, LPCTSTR
szLayerName = NULL);
```

Purpose

Creates a Circle in specified Layer or current Layer, if Layer name is omitted.

Parameters

`double dSx`, `double dSy`: The center of Circle;

`RCEAPoint ptCenter`: The center of Circle;

`double dRadius`: Radius;

`LPCTSTR szLayerName`: Name of the layer.

Return

`HOBJECT`: Handle to created circle; `EA_INVALID_OBJECT` if failed.

Example

```
HOBJECT hObject = ExpApiCreateCircle (0, 0, 100, "Layer_1");
```

18.7.16: Ellipse

```
HOBJECT ExpApiCreateEllipse(double dSx, double dSy, double dEx, double dEy,
LPCTSTR szLayerName = NULL);
HOBJECT ExpApiCreateEllipse(RCEAPoint ptCenter, RCEAPoint ptRadius, LPCTSTR
szLayerName = NULL);
HOBJECT ExpApiCreateEllipse(RCEARect rtRect, LPCTSTR szLayerName = NULL);
```

Purpose

Creates an Ellipse in specified Layer or current Layer, if Layer name is omitted.

Parameters

`double dSx`, `double dSy`: The center of Ellipse;

`RCEAPoint ptCenter`: The center of Ellipse;

`double dEx`, `double dEy`: Radius 1, Radius 2;

`RCEAPoint ptRadius`: Radius 1, Radius 2;

`RCEARect rtRect`: The center of Ellipse, Radius 1, Radius 2;

`LPCTSTR szLayerName`: Name of the Layer.

Return

`HOBJECT`: Handle to created ellipse; `EA_INVALID_OBJECT` if failed.

Example

```
HOBJECT hObject = ExpApiCreateEllipse (0, 0, 100, 200, "Layer_1");
```

18.7.17: Donut

```
HOBJECT ExpApiCreateDonut(double dSx, double dSy, double dRad1, double dRad2,
LPCTSTR szLayerName = NULL);
HOBJECT ExpApiCreateDonut(RCEAPoint ptCenter, double dRad1, double dRad2,
LPCTSTR szLayerName = NULL);
```

Purpose

Creates a Donut in specified Layer or current Layer, if Layer name is omitted.

Parameters

`double dSx`, `double dSy`: The center of Donut;

`RCEAPoint ptCenter`: The center of Donut;

`double dRad1`, `double dRad2`: Radius 1, Radius 2;

`LPCTSTR szLayerName`: Name of the Layer.

Return

`HOBJECT`: Handle to created donut; `EA_INVALID_OBJECT` if failed.

Example

```
HOBJECT hObject = ExpApiCreateDonut (CEAPoint(0, 0), 100, 200, "Layer_1");
```

18.7.18: Wire

```
HOBJECT ExpApiCreateWire(RCEAPointsArray Points, double dWidth, LPCTSTR
szLayerName = NULL);
```

Purpose

Creates a Wire in specified Layer or current Layer, if Layer name is omitted.

Parameters

`RCEAPointsArray Points`: Reference to array of Wire coordinates;

`double dWidth`: Wire width;

`LPCTSTR szLayerName`: Name of the Layer.

Return

HOBJECT: Handle to created Wire; EA_INVALID_OBJECT if failed.

Example

```
CEAPointsArray WirePoints;
WirePoints.Add(CEAPoint(1000.0, 2000.0));
WirePoints.Add(CEAPoint(2000.0, 1000.0));
HOBJECT hObject = ExpApiCreateWire(WirePoints, 20.0, "Layer_1");
```

18.7.19: Region

```
HOBJECT ExpApiCreateRegion(RCEAPointsArray Points, LPCTSTR szLayerName =
NULL)
```

Purpose

Creates a Region in specified Layer or current Layer, if Layer name is omitted.

Parameters

RCEAPointsArray Points: Reference to array of Region coordinates;

LPCTSTR szLayerName: Name of the Layer.

Return

HOBJECT: Handle to created region; EA_INVALID_OBJECT if failed.

Example

```
CEAPointsArray RegPoints;
RegPoints.Add(CEAPoint(1000.0, 2000.0));
RegPoints.Add(CEAPoint(2000.0, 1000.0));
RegPoints.Add(CEAPoint(1000.0, 1000.0));
HOBJECT hObject = ExpApiCreateRegion(RegPoints, "Layer_1");
```

18.7.20: Text

```
HOBJECT ExpApiCreateText(LPCTSTR sText, double dSx, double dSy, double
dtWidth, double dtHeight, double dEscapement = 0., BOOL bSlant = FALSE, BOOL
bMirroring = FALSE, LPCTSTR szLayerName = NULL);
HOBJECT ExpApiCreateText(LPCTSTR szText, RCEAPoint ptPos, double dWidth,
double dHeight, double dEscapement = 0., BOOL bSlant = FALSE, BOOL bMirroring
= FALSE,                                  LPCTSTR szLayerName = NULL);
HOBJECT ExpApiCreateText(LPCTSTR szText, RCEARect rtRect, double dEscapement
= 0., BOOL bSlant = FALSE, BOOL bMirroring = FALSE, LPCTSTR szLayerName =
NULL);
```

Purpose

Creates a Text in specified Layer or current Layer, if Layer name is omitted.

Parameters

double dSx, double dSy: Initial placement position of Text;

RCEAPoint ptPos: Initial placement position of Text;

double dWidth, double dHeight: Text width, Text height;

`RCEARect rtRect`: Initial placement position, Text width, Text height

`double dEscapement`: Escapement of Text

`BOOL bSlant`: Slanted Text;

`BOOL bMirroring`: Mirrored Text;

`LPCTSTR szLayerName`: Name of the Layer.

Return

`HOBJECT`: Handle to created text; `EA_INVALID_OBJECT` if failed.

Example

```
HOBJECT hObject =ExpApiCreateText("Some Text", CEAPoint(10.0, 20.0), 100,
100, 0.0, FALSE, TRUE, "Layer_1");
```

18.7.21: Link Text

```
BOOL ExpApiLinkText(HOBJECT hObject, HOBJECT hText);
```

Purpose

Links text to object.

Parameters

`HOBJECT hObject`: Handle of Object;

`HOBJECT hText`: Handle of text object;

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
HOBJECT hObject = ...some object...;
HOBJECT hText = ExpApiCreateText("This is attached text", CEAPoint(0.0,
10.0), 100, 100, 0.0, FALSE, TRUE);
BOOL bResult = ExpApiLinkText(hObject, hText);
```

18.7.22: Unlink Text

```
HOBJECT ExpApiUnlinkText(HOBJECT hObject, int nText);
```

Purpose

Unlinks the linked text, given its number, from the specified object then inserts it into the layout and returns its handle.

Parameters

`HOBJECT hObject`: Handle to object;

`int nText`: Linked text zero-based index.

Return

`HOBJECT`: The handle to unlinked text; `EA_INVALID_OBJECT` if failed.

Example

```
HOBJECT hObject =...some object...;
int nText = 0; //Linked Text number
HOBJECT hText = ExpApiUnlinkText(hObject, nText);
```

18.7.23: Add Linked Text

```
BOOL ExpApiAddLinkedText(HOBJECT hObject, RCEAPoint Position, double
dHeight, double dWidth, LPCTSTR szText);
```

Purpose

Creates and adds linked text to object.

Parameters

`HOBJECT hObject`: Handle of Object;

`RCEAPoint ptPos`: Initial placement position of Text;

`double dWidth`, `double dHeight`: Text width, Text height;

`LPCTSTR szText`: Text to attach.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
HOBJECT hObject = …some object…;
BOOL bResult = ExpApiAddLinkedText(hObject, CEAPoint(0.0, 10.0), 100, 100,
szText);
```

18.7.24: Delete Linked Text

```
BOOL ExpApiDeleteLinkedText(HOBJECT hObject, int nText);
```

Purpose

Deletes the Linked Text, given it's number, from specified Object.

Parameters

`HOBJECT hObject`: Handle to Object;

`int nText`: Linked text zero-based index.

Return

`BOOL`: TRUE, if Linked Text is deleted successfully; FALSE, if failed.

Example

```
HOBJECT hObject = …some object…;
int nText = 0; //Linked Text number
BOOL bResult = ExpApiDeleteLinkedText(hObject, nText);
```

18.7.25: Delete All Linked Text

```
HOBJECT ExpApiDeleteLinkedTexts(HOBJECT hObject);
```

Purpose

Deletes all Linked Texts from specified Object.

Parameters

`HOBJECT hObject`: Handle to Object.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
HOBJECT hObject = …some object…;
BOOL bResult = ExpApiDeleteLinkedTexts(hObject);
```

18.7.26: Get Linked Texts Number

```
int ExpApiGetLinkedTextsNumber(HOBJECT hObject);
```

Purpose

Retrieves the number of Linked Texts attached to the specified Object.

Parameters

`HOBJECT hObject`: Handle to Object.

Return

`int`: Non-negative number of Linked Texts, if successful; negative, if failed.

Example

```
HOBJECT hObject = …some object…;
int nTexts = ExpApiGetLinkedTextsNumber(hObject);
```

18.7.27: Get Linked Text

```
BOOL ExpApiGetLinkedText(HOBJECT hObject, int nText, OBJECT_PROPERTY&
Property);
```

Purpose

Retrieves the Linked Text, given it's number, attached to the specified Object.

Parameters

`HOBJECT hObject`: Handle to Object;

`int nText`: Linked text zero-based index

`OBJECT_PROPERTY& Property`: Reference to Linked Text property retrieved.

Return

`BOOL`: TRUE, if Linked Text is retrieved successfully; FALSE, if failed.

Example

```
HOBJECT hObject = …some object…;
int nText = 0; // Linked Text number
OBJECT_PROPERTY Property;
```

```
BOOL bIsOk = ExpApiGetLinkedText(hObject, nText, Property);
```

18.7.28: Add User-Defined Property

```
BOOL ExpApiAddUserProperty (HOBJECT hObject, LPCTSTR szName, LPCTSTR
szValue);
```

Purpose

Adds User-Defined Property with specified Name and Value to Object.

Parameters

`HOBJECT hObject`: Handle of Object;

`LPCTSTR szName`: Property Name;

`LPCTSTR szValue`: Property Value.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
HOBJECT hObject = …some object…;
BOOL bResult = ExpApiAddUserProperty (hObject, "property17", "17");
```

18.7.29: Get User-Defined Property

```
LPTSTR ExpApiGetUserProperty(HOBJECT hObject, LPCTSTR szName);
```

Purpose

Retrieves Value of Object's User-Defined Property with specified Name.

`ExpApiGetProperty()` allocates memory for Value, and this memory should be explicitly freed later, see the example.

Parameters

`HOBJECT hObject`: Handle of Object.

`LPCTSTR szName`: Property Name.

Return

`LPCTSTR szValue`: Property Value.

Example

```
HOBJECT hObject = …some object…;
LPTSTR szValue = ExpApiGetUserProperty(hObject, "property17");
….(do something)
if (szValue)
     free(szValue);
```

18.7.30: Modify User-Defined Property

```
BOOL ExpApiModifyUserProperty (HOBJECT hObject, LPCTSTR szName, LPCTSTR
szValue);
```

Purpose

Modifies Value of Object's User-Defined Property with specified Name. Memory allocated by `ExpApiModifyProperty()` should be finally freed.

Parameters

`HOBJECT hObject`: Handle of Object;

`LPCTSTR szName`: Property Name;

`LPCTSTR szValue`: Property Value to set.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
HOBJECT hObject = …some object…;
BOOL bResult = ExpApiModifyUserProperty (hObject, "property17", "17");
```

18.7.31: Delete User-Defined Property

```
BOOL ExpApiDeleteUserProperty(HOBJECT hObject, LPCTSTR szName);
```

Purpose

Deletes Object's User-Defined Property with specified Name.

Parameters

`HOBJECT hObject`: Handle of Object;

`LPCTSTR szName`: Property Name.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
HOBJECT hObject = …some object…;
BOOL bResult = ExpApiDeleteUserProperty(hObject, "property17");
```

18.7.32: Delete All User-Defined Properties

```
BOOL  ExpApiDeleteUserPropertyAll(HOBJECT hObject);
```

Purpose

Deletes All Object's User-Defined Properties.

Parameters

`HOBJECT hObject`: Handle of Object.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
HOBJECT hObject = …some object…;
BOOL bResult = ExpApiDeleteUserProperty(hObject, "property17");
```

18.8: Tools

18.8.1: Merge Wires

```
HOBJECT ExpApiMergeWires(HOBJECT hObject1, BOOL bFirst1, HOBJECT hObject2,
BOOL bFirst2, LPCTSTR szCellName = NULL);
```

Purpose

Merges two separate wires into one wire in specified cell or current cell, if cell name is omitted.

Parameters

`HOBJECT hObject1`: The first wire;

`BOOL bFirst1`: TRUE if the first vertex of the first wire connects to the second wire; FALSE if the last vertex does;

`HOBJECT hObject2`: The second wire;

`BOOL bFirst2`: TRUE if the first vertex of the second wire connects to the first wire; FALSE if the last vertex does;

`LPCTSTR szCellName`: Name of the cell.

Return

`HOBJECT`: Resultant wire.

Example

```
HOBJECT hObject = ExpApiMergeWires(hObject1, TRUE, hObject2, TRUE, "Cell_1")
```

18.8.2: Merge Selected

```
BOOL ExpApiMergeSelected();
```

Purpose

Produces the union of shape objects selected in the active cell.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiMergeSelected();
```

18.8.3: Resize Selected

```
BOOL ExpApiResizeSelected(double dSize, BOOL bUnderSize = FALSE, BOOL bGrid =
FALSE, BOOL bMerge = FALSE);
```

Purpose

Resizes the selected shape objects by the given value.

Parameters

`double dSize`: Defines the shape shift size in current measurement units;

`BOOL bUnderSize`: UnderSize (TRUE to shrinks the covered areas) or OverSize (FALSE to expand the covered areas);

`BOOL bGrid`: Flag to grid put the resized shapes on minimal grid;

`BOOL bMerge`: Flag to merge the resized shapes.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiResizeSelected(100, FALSE, FALSE, FALSE);
```

18.8.4: Explode Selected

```
BOOL ExpApiExplodeSelected();
```

Purpose

Explodes the selected cell instance/array Objects.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiExplodeSelected();
```

18.8.5: Explode All

```
BOOL ExpApiExplodeAll();
```

Purpose

Explodes all cell instance/array Objects. Decreases cell hierarchy depth by one.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiExplodeAll();
```

18.8.6: Flatten

```
BOOL ExpApiFlatten(LPCTSTR szFileName, LPCTSTR szCellName = NULL);
```

Purpose

Creates a new flat project and saves it the specified file in eld format, from the specified cell or current cell, if cell name is omitted.

Parameters

`LPCTSTR szFileName`: Output `*.eld` file name;

`LPCTSTR szCellName`: Cell name.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiFlatten("FlattenCell_1.eld", "Cell_1");
```

18.8.7: Cut by Line

```
BOOL ExpApiCutByLine(RCEAPointsArray pLinePoints, LPCTSTR szCellName =
NULL);
```

Purpose

Cuts objects by Line in specified cell or current cell, if cell name is omitted.

Parameters

`RCEAPointsArray pLinePoints`: Vertices of Line;

`LPCTSTR szCellName`: Cell name.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
CEAPointsArray LinePoints;
LinePoints.Add(CEAPoint(-1000.0, 0.0));
LinePoints.Add(CEAPoint(0.0, 0.0));
LinePoints.Add(CEAPoint(0.0, 2000.0));
BOOL bResult = ExpApiCutByLine(LinePoints, "cell_1");
```

18.8.8: Set Cut by Line Mode

```
BOOL ExpApiSetCutByLineMode(int nCutMode);
```

Purpose

Sets Cut by Line Mode (see table Cut by Line Modes (`CUT_MODE`)).

Parameters

`int nCutMode`: Cut by Line Mode.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetCutByLineMode(CM_SELECTED); // cut only selected
objects by line
BOOL bResult = ExpApiSetCutByLineMode(CM_CURRENT); // cut only objects on
active layer
```

```
BOOL bResult = ExpApiSetCutByLineMode(CM_ALL); // cut all objects by line
```

18.8.9: Set Cut by Line Wire Mode

```
BOOL ExpApiSetCutByLineWireMode(BOOL bSaveWire);
```

Purpose

Sets Cut by Line Wire Mode to specified.

Parameters

`BOOL bSaveWire`: Cut by Line Wire Mode (TRUE to save wire; FALSE otherwise).

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetCutByLineWireMode(TRUE); // save wire
BOOL bResult = ExpApiSetCutByLineWireMode(FALSE); // do not save wire
```

18.8.10: Get Cut by Line Mode

```
void ExpApiGetCutByLineMode(int* nCutMode, BOOL* bSaveWire);
```

Purpose

Retrieves Cut by Line Mode and Wire Mode (see table Cut by Line Modes (`CUT_MODE`)).

Parameters

`int* nCutMode`: Retrieved Cut by Line Mode;

`BOOL* bSaveWire`: Retrieved Cut by Line Wire Mode.

Return

None

Example

```
int nCutMode;
BOOL bSaveWire;
ExpApiGetCutByLineMode(&nCutMode, &bSaveWire);
```

18.8.11: Wiring

```
HOBJECT ExpApiRunWiring(RCEAPoint ptPoint1, RCEAPoint ptPoint2, RCEARect
rtArea, double dWidth = 0, double dSpace = 0, LPCTSTR szLayerlName = NULL);
```

Purpose

To create the shortest wire possible to connect two points.

Parameters

`RCEAPoint ptPoint1`: First point.

`RCEAPoint ptPoint2`: Second point.

`RCEARect rtArea`: Routing area. If omitted, it will be defined automatically.

`double dWidth`: Wire width. If omitted, the width from the layer definition will be used.

`double dSpace`: Spacing between wire and obstacles. If omitted, zero spacing will be used.

`LPCTSTR szLayerName`: Name of Layer. If omitted, the active layer will be used.

Return

`HOBJECT`: Resultant wire.

Example

```
HOBJECT hWire = ExpApiRunWiring(CEAPoint(0.0, 0.0), CEAPoint(100.0, 100.0));
```

18.9: Groups

18.9.1: Create Group

```
BOOL ExpApiGroupsCreate();
```

Purpose

Creates a group from selected objects.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiGroupsCreate();
```

18.9.2: Remake Groups

```
BOOL ExpApiGroupsRemake();
```

Purpose

Destroys selected groups and creates new group from selected objects.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiGroupsRemake();
```

18.9.3: Destroy Groups

```
BOOL ExpApiGroupsDestroy();
```

Purpose

Destroys selected groups.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiGroupsDestroy();
```

18.9.4: Destroy All Groups

```
BOOL ExpApiGroupsDestroyAll();
```

Purpose

Destroys all groups.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiGroupsDestroyAll();
```

18.10: Selection Operations

18.10.1: Select Newest

```
BOOL ExpApiSelectNewest();
```

Purpose

Selects object(s) created by the last editing operation.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSelectNewest();
```

18.10.2: Select Object

```
BOOL ExpApiSelectObject(HOBJECT hObject);
```

Purpose

Selects Object.

Parameters

`HOBJECT hObject`: Handle of Object to select.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
HOBJECT hObject = … some object … ;
BOOL bResult = ExpApiSelectObject (hObject);
```

18.10.3: Select Layer

```
BOOL ExpApiSelectLayer(LPCTSTR szLayerName = NULL);
```

Purpose

Selects all objects in specified Layer or current Layer, if Layer name is omitted.

Parameters

`LPCTSTR szLayerName`: Name of the Layer.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSelectLayer("Layer_1");
```

18.10.4: Select All

```
BOOL ExpApiSelectAll();
```

Purpose

Selects all objects in current cell.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSelectAll();
```

18.10.5: Deselect All

```
BOOL ExpApiDeselectAll();
```

Purpose

Deselects all objects in current cell.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiDeselectAll();
```

18.10.6: Invert Selection

```
BOOL ExpApiInvertSelection();
```

Purpose

Inverts selection of all objects in current cell.

Parameters

None

Returns

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiInvertSelection();
```

18.10.7: Select by Type

```
BOOL ExpApiSelectByType(SEARCH_OBJECT_TYPE sType);
```

Purpose

Selects all Objects of specified Type in current cell.

Parameters

`SEARCH_OBJECT_TYPE sType`: Type of object to select (see List of Structures for structure description).

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSelectByType(SOT_BOX);
```

18.10.8: Select by Box

```
BOOL ExpApiSelectByBox(double dSx, double dSy, double dEx, double dEy,
SELECT_BOOL_MODE selMode = SBM_SELECT_LAST);
BOOL ExpApiSelectByBox(RCEAPoint ptStart, RCEAPoint ptEnd, SELECT_BOOL_MODE
selMode = SBM_SELECT_LAST);
BOOL ExpApiSelectByBox(RCEARect rtRect, SELECT_BOOL_MODE selMode =
SBM_SELECT_LAST);
```

Purpose

Selects all Objects by the specified box region.

Parameters

`double dSx`, `double dSy`: Start point of Selection region;

`double dEx`, `double dEy`: End point of Selection region;

`RCEAPoint ptStart`: Start point of Selection region;

`RCEAPoint ptEnd`: End point of Selection region;

`RCEARect rtRect`: Start and end points of Selection region;

SELECT_BOOL_MODE selMode: Selection mode, or current mode if it is omitted.

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSelectByBox(0.0, 0.0, 100.0, 100.0);
```

18.11: Selection Modes

18.11.1: Set Select Mode

BOOL ExpApiSetSelectMode(SELECT_BOOL_MODE Mode);

Purpose

Sets the selection mode.

Parameters

SELECT_BOOL_MODE Mode

SBM_SELECT_SELECT: The picked objects are added to the selected set.

SBM_SELECT_RESELECT: The previously selected objects are deselected, and the picked objects become selected.

SBM_SELECT_DESELECT: The picked objects are removed from the selected set.

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetSelectMode(SBM_SELECT_RESELECT);
```

18.11.2: Get Select Mode

```
SELECT_BOOL_MODE     ExpApiGetSelectMode();
```

Purpose

Gets current selection mode (See Section 18.11.1: “Set Select Mode”).

Parameters

None

Return

SELECT_BOOL_MODE: Current selection mode.

Example

```
SELECT_BOOL_MODE Mode = ExpApiGetSelectMode();
```

18.11.3: Set Concern Mode

```
BOOL ExpApiSetConcernMode(SELECT_CONCERN_MODE Mode);
```

Purpose

Sets selection concern mode.

Parameters

`SELECT_CONCERN_MODE Mode`.

`SCM_INSIDE`: Select the objects that fit completely inside the selection box.

`SCM_TOUCH`: Select the objects that cross the selection box.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetConcernMode (SCM_INSIDE);
```

18.11.4: Get Concern Mode

```
SELECT_CONCERN_MODE ExpApiGetConcernMode();
```

Purpose

Gets current selection concern mode (See Section 18.11.3: "Set Concern Mode").

Parameters

None

Return

`SELECT_CONCERN_MODE`: Current concern mode

Example

```
SELECT_CONCERN_MODE Mode = ExpApiGetConcernMode();
```

18.11.5: Set Fill Mode

```
BOOL ExpApiSetFillMode(SELECT_FILL_MODE Mode);
```

Purpose

Sets selection fill mode.

Parameters

`SELECT_FILL_MODE Mode`

`SFM_BOUNDARY`: In this mode, objects are (de)selectable by any of its boundary points.

`SFM_SOLID`: In this mode, objects are (de)selectable by any of its inside or boundary points.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetFillMode(SFM_SOLID);
```

18.11.6: Get Fill Mode

```
SELECT_FILL_MODE ExpApiGetFillMode();
```

Purpose

Gets selection fill mode (See Section 18.11.5: “Set Fill Mode”).

Parameters

None

Return

`SELECT_FILL_MODE`: Current fill mode

Example

```
SELECT_FILL_MODE Mode = ExpApiGetFillMode();
```

18.11.7: Set Ignore Instances

```
BOOL ExpApiSetIgnoreInstances(BOOL bMode = TRUE);
```

Purpose

Switches off/on the selectivity of cell instances/arrays in the layout.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetIgnoreInstances ();
```

18.11.8: Get Ignore Instances

```
BOOL ExpApiGetIgnoreInstances ();
```

Purpose

Gets current selectivity of cell instances/arrays in the layout.

Parameters

None

Return

`BOOL`: Current selectivity of cell instances/arrays in the layout.

Example

```
BOOL bResult = ExpApiGetIgnoreInstances ();
```

18.11.9: Set Ignore Primitives

```
BOOL ExpApiSetIgnorePrimitives(BOOL bMode = TRUE);
```

Purpose

Switch off/on the selectivity of primitive objects in the layout.

Parameters

None

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiSetIgnorePrimitives ();
```

18.11.10: Get Ignore Primitives

```
BOOL ExpApiGetIgnorePrimitives();
```

Purpose

Gets current selectivity of primitive objects in the layout.

Parameters

None

Return

BOOL: Current selectivity of primitive objects in the layout.

Example

```
BOOL bResult = ExpApiGetIgnorePrimitives ();
```

18.12: Edit Operations

18.12.1: Cut

```
BOOL ExpApiCut()
```

Purpose

Cuts selected cell objects to clipboard.

Parameters

None

Return

BOOL: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiCut();
```

18.12.2: Copy

```
BOOL ExpApiCopy();
```

Purpose

Copies selected cell objects to clipboard.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiCopy();
```

18.12.3: Paste

```
BOOL ExpApiPaste(double dx, double dy);
BOOL ExpApiPaste(RCEAPoint pt);
```

Purpose

Pastes current clipboard contents into new place with placement coordinates.

Parameters

`double dx`, `double dy`: Placement coordinates;

`RCEAPoint pt`: Placement coordinates.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiPaste(50.0, 80.0);
BOOL bResult = ExpApiPaste(CEAPoint(100.0, 200.0));
```

18.12.4: Move

```
BOOL ExpApiMove(double dx, double dy);
BOOL ExpApiMove(RCEAPoint pt);
```

Purpose

Moves selected cell objects to new place shifted relatively to the initial position.

Parameters

`double dx`, `double dy`: Placement coordinates;

`RCEAPoint pt`: Placement coordinates.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiMove(50.0, 80.0);
BOOL bResult = ExpApiMove(CEAPoint(100.0, 200.0));
```

18.12.5: Duplicate

```
BOOL ExpApiDuplicate (double dx, double dy);
BOOL ExpApiDuplicate(RCEAPoint pt);
```

Purpose

Duplicates selected cell objects to a place shifted relatively to the initial position.

Parameters

`double dx`, `double dy`: Placement coordinates;

`RCEAPoint pt`: Placement coordinates.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiDuplicate(50.0, 80.0);
BOOL bResult = ExpApiDuplicate(CEAPoint(100.0, 200.0));
```

18.12.6: Rotate

```
BOOL ExpApiRotate(double dx, double dy, double dAngle, BOOL bDuplicate =
FALSE);
BOOL ExpApiRotate(RCEAPoint pt, double dAngle, BOOL bDuplicate = FALSE);
```

Purpose

Rotates selected cell objects by a given angle and at specific position with or without duplication.

Parameters

`double dx`, `double dy`: Placement coordinates;

`RCEAPoint pt`: Placement coordinates;

`double dAngle`: Rotation angle;

`bDuplicate`: Duplicate flag.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiRotate(50.0, 80.0, 45); // without duplication
BOOL bResult = ExpApiRotate(CEAPoint(100.0, 200.0), 45, TRUE);
```

18.12.7: Mirror Vertical

```
BOOL ExpApiMirrorVertical(double dAxis, BOOL bDuplicate = FALSE);
```

Purpose

Vertically flips selected cell objects with respect to the specified horizontal axis.

Parameters

`double dAxis`: Y-coordinate of the horizontal axis.

`bDuplicate`: Duplicate flag.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiMirrorVertical(100.0);
```

18.12.8: Mirror Horizontal

```
BOOL ExpApiMirrorHorizontal(double dAxis, BOOL bDuplicate = FALSE);
```

Purpose

Horizontally flips selected cell objects with respect to the specified vertical axis.

Parameters

`double dAxis`: X-coordinate of the vertical axis.

`bDuplicate`: Duplicate flag.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiMirrorHorizontal(100.0);
```

18.12.9: Move To Active Layer

```
BOOL ExpApiMoveToActiveLayer();
```

Purpose

Moves selected primitive Objects to active Layer.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiMoveToActiveLayer();
```

18.12.10: Copy To Active Layer

```
BOOL ExpApiCopyToActiveLayer();
```

Purpose

Copies selected primitive Objects to active Layer.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiCopyToActiveLayer();
```

18.12.11: Delete Selected Objects

```
BOOL ExpApiDeleteSelected();
```

Purpose

Deletes selected Objects.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
BOOL bResult = ExpApiDeleteSelected();
```

18.12.12: Replace Selected Instances

```
BOOL ExpApiReplaceSelectedInstances(LPCTSTR szTargetCellName, LPCTSTR
szCellName = NULL);
```

Purpose

Replaces selected Instances by Instances of TargetCell inside the specified cell, or current cell if cell name is omitted.

Parameters

`LPCTSTR szTargetCellName`: Target cell name;

`LPCTSTR szCellName`: Cell name;

Return

`BOOL`: TRUE if successful; FALSE otherwise

Example

```
BOOL bResult = ExpApiReplaceSelectedInstances("Target_Cell");
BOOL bResult = ExpApiReplaceSelectedInstances("Target_Cell", "Cell_1");
```

18.12.13: Replace Instances in Cell

```
BOOL ExpApiReplaceInstancesInCell(LPCTSTR szSourceCellName, LPCTSTR
szTargetCellName, LPCTSTR szCellName = NULL);
```

Purpose

Replaces Instances of SourceCell by Instances of TargetCell inside the specified cell, or current cell if cell name is omitted.

Parameters

`LPCTSTR szSourceCellName`: Source cell name;

`LPCTSTR szTargetCellName`: Target cell name;

`LPCTSTR szCellName`: Work cell name;

Return

BOOL: TRUE if successful; FALSE otherwise

Example

```
BOOL bResult = ExpApiReplaceInstancesInCell("Source_Cell", "Target_Cell");
BOOL bResult = ExpApiReplaceInstancesInCell("Source_Cell", "Target_Cell",
"Cell_1");
```

18.12.14: Replace Instances in Project

```
BOOL ExpApiReplaceInstancesInProject (LPCTSTR szSourceCellName, LPCTSTR
szTargetCellName);
```

Purpose

Replaces Instances of SourceCell by Instances of TargetCell in the whole project.

Parameters

LPCTSTR szSourceCellName: Source cell name;

LPCTSTR szTargetCellName: Target cell name;

Return

BOOL: TRUE if successful; FALSE otherwise

Example

```
BOOL bResult = ExpApiReplaceInstancesInProject("Source_Cell",
"Target_Cell");
```

18.13: Support

18.13.1: File Name Selector

```
LPTSTR ExpApiGetFileName(LPCTSTR sFileName, LPCTSTR lpszFilter = NULL,
LPCTSTR lpszDefExt = NULL, BOOL bLoad = TRUE)
```

Purpose

Selects file name. Memory allocated by the function should be finally freed.

Parameters

LPCTSTR szFileName: File name;

LPCTSTR lpszFilter: Filter;

LPCTSTR lpszDefExt: Default extension;

BOOL bLoad: Flag (TRUE calls the **Load** dialog, FALSE calls the **Save As** dialog).

Return

LPTSTR: Path to selected file.

Example

```
LPTSTR szFilePath = ExpApiGetFileName("Project_1", "Expert Files
(*.eld)|*.eld|", "eld", TRUE);
```

```
if (szFilePath)
{
free(szFilePath);
}
```

18.13.2: Update View

```
void ExpApiUpdate();
```

Purpose

Updates current contents of screen (useful for Debug).

Parameters

None

Return

None

Example

```
ExpApiUpdate();
```

18.13.3: Refresh View

```
void ExpApiRefreshView(CEARect cptRect = EA_INITIAL_RECT);
```

Purpose

Refreshes current View in specified rectangle or all view, if rectangle is omitted.

Parameters

`CEARect cptRect`: View rectangle to refresh.

Return

None

Example

```
ExpApiRefreshView();
ExpApiRefreshView(CEARect(10.0, 10.0, 100.0, 100.0));
```

18.13.4: Refresh Object

```
void ExpApiRefreshObject(HOBJECT hObject);
```

Purpose

Refreshes specified Object.

Parameters

`HOBJECT`: Handle of Object to refresh.

Return

None

Example

```
HOBJECT hObject = … some object …;
ExpApiRefreshObject(hObject);
```

18.13.5: Get Error Code

```
DWORD ExpApiGetErrorCode();
```

Purpose

Retrieves API Error Code (see table API Error Codes and Messages).

Parameters

None

Return

`DWORD`: API Error Code.

Example

```
DWORD dwErrorCode = ExpApiGetErrorCode();
```

18.13.6: Get Error Message

```
LPCTSTR ExpApiGetErrorMessage();
```

Purpose

Retrieves API Error Message (see table API Error Codes and Messages).

Parameters

None

Return

`LPCTSTR`: API Error Message.

Example

```
LPCTSTR szErrorMessage = ExpApiGetErrorMessage();
```

18.14: Script Language (LISA)

18.14.1: XI Initialization

```
void ExpApiXiInit();
```

Purpose

Initializes LISA scripting tool. Loads predefined Expert commands.

Parameters

None

Return

None

Example

```
ExpApiXiInit();
```

18.14.2: XI Run

```
BOOL ExpApiXiRun(LPCTSTR sz);
```

Purpose

Runs LISA script or command.

Parameters

`LPCTSTR`: script string to execute.

Return

`BOOL`: TRUE if successful; FALSE otherwise.

Example

```
ExpApiXiRun("run script.xis");
ExpApiXiRun("box 0 0 100 100");
```

18.15: User Interface (UI)

18.15.1: Get Mouse Point

```
BOOL ExpApiGetMousePoint(CEAPoint& pt, RCEAPoint ptPrev  = EA_MAX_POINT);
```

Purpose

Reads out mouse cursor coordinates.

Parameters

`RCEAPoint ptPrev`: Previous point. If this parameter is given and not equal to `EA_MAX_POINT`, then a line is drawn from ptPrev to the current mouse position.

`CEAPoint& pt`: Output mouse cursor coordinates.

Return

`BOOL`: TRUE, if mouse cursor coordinates are valid; FALSE otherwise.

Example

```
// draws the line from ptPrev to current point
CEAPoint ptPrev = CEAPoint(100, 200);
CEAPoint pt;
BOOL bValid = ExpApiGetMousePoint(pt, ptPrev);
// does not draw line
CEAPoint pt;
BOOL bValid = ExpApiGetMousePoint(pt);
```

18.15.2: Get Mouse Rectangle

```
BOOL ExpApiGetMouseRect(CEARect& r);
```

Purpose

Reads and stores coordinates of mouse tracking rectangle.

Parameters

`CEARect& r`: Output coordinates of mouse tracking rectangle.

Return

`BOOL`: TRUE, if mouse cursor coordinates are valid; FALSE otherwise.

Example

```
CEARect mRect;
BOOL bValid = ExpApiGetMouseRect(mRect);
```

18.15.3: Is Mouse Right Button Pressed

```
BOOL ExpApiIsRMousePress();
```

Purpose

Catches right mouse button press event.

Parameters

None

Return

`BOOL`: TRUE if pressed; FALSE otherwise.

Example

```
OBJECT_PROPERTY Property;
CEAPoint ptPrev = EA_MAX_POINT;
CEAPoint pt;
do
 {
    pt = ExpApiGetMousePoint(ptPrev);
    Property.VerticesArray.Add(pt);
    ptPrev = pt;
 }
 while (!ExpApiIsRMousePress());
```

18.16: Library

18.16.1: Set Logical Name

```
BOOL ExpApiSetLogName(LPCTSTR szLibLogName, LPCTSTR szLibPathName);
```

Purpose

Sets Library logical name for specified Library path in the Library Name Table.

Parameters

`LPCTSTR szLibLogName`: Library logical name;

`LPCTSTR szLibPathName`: Library file pathname.

Return

BOOL: TRUE if successful; FALSE otherwise

Example

```
BOOL bResult = ExpApiSetLogName("library1", "testlibrary1.eld");
```

18.16.2: Get Logical Name By File Name

```
LPCTSTR ExpApiGetLogNameByFileName(LPCTSTR szLibPathName);
```

Purpose

Gets Library logical name by specified Library path.

Parameters

LPCTSTR szLibPathName: Library path.

Return

LPCTSTR: Library logical name.

Example

```
LPCTSTR szLibLogName = ExpApiGetLogNameByFileName("testlibrary1.eld");
```

18.16.3: Get File Name By Logical Name

```
LPCTSTR ExpApiGetFileNameByLogName(LPCTSTR szLibLogName);
```

Purpose

Gets Library path by specified Library logical name.

Parameters

LPCTSTR szLibLogName: Library logical name.

Return

LPCTSTR: Library path.

Example

```
LPCTSTR szLibPathName = ExpApiGetFileNameByLogName("library1");
```

18.16.4: Activate Library

```
BOOL ExpApiLibActivate(LPCTSTR szLibLogName);
```

Purpose

Activates Library with specified logical name.

Parameters

LPCTSTR szLibLogName: Library logical name.

Return

BOOL: TRUE if successful; FALSE otherwise

Example

```
BOOL bResult = ExpApiLibActivate("library1");
```

18.16.5: Deactivate Library

```
BOOL ExpApiLibDeactivate(LPCTSTR szLibLogName);
```

Purpose

Deactivates Library with specified logical name.

Parameters

`LPCTSTR szLibLogName`: Library logical name.

Return

`BOOL`: TRUE if successful; FALSE otherwise

Example

```
BOOL bResult = ExpApiLibDeactivate("library1");
```

18.16.6: Delete Library

```
BOOL ExpApiLibDelete(LPCTSTR szLibLogName);
```

Purpose

Deletes Library with specified logical name from the Library Name Table.

Parameters

`LPCTSTR szLibLogName`: Library logical name.

Return

`BOOL`: TRUE if successful; FALSE otherwise

Example

```
BOOL bResult = ExpApiLibDelete("library1");
```

18.16.7: Delete All Libraries

```
BOOL ExpApiLibDeleteAll();
```

Purpose

Deletes All Libraries from the Library Name Table.

Parameters

None

Return

`BOOL`: TRUE if successful; FALSE otherwise

Example

```
BOOL bResult = ExpApiLibDeleteAll();
```

18.16.8: Get library Path Name

```
LPCTSTR ExpApiGetLibraryPathName(LPCTSTR szLogName = NULL)
```

Parameters

`LPCTSTR szLogName`: Logical library name. If `szLogName` is NULL of empty("") then this function gets the pathname for the current project file.

Return

`LPCTSTR`: Full path to library.

18.16.9: Get library Directory Name

```
LPCTSTR ExpApiGetLibraryDirName(LPCTSTR szLogName = NULL)
```

Parameters

`LPCTSTR szLogName`: Library name. If `szLogName` is NULL of empty("") then this function gets the pathname for the current project file.

Return

`LPCTSTR`: Directory name of library

Index

A

active layer 3-6, 5-9
Activity log 11-11
Activity Log, Recovery Log 17-121
admissible angles 5-2
ancestor 2-2
Angle modes 5-2
apl format 2-8
Apple format 2-8
Applicon 4-7, 11-42
Applicon file 2-8
array 2-1, 2-6
array delta 2-1
array instance 2-1
array position 2-6
array step 2-1
autopan 9-7
Autorename 4-17

B

Backup name 4-6
Basic Notions 2-1
Basics of Usage in Expert 14-2
bindkey 3-14, 11-44
Bindkeys instead of mouse clicks 3-12
bounding box 2-1
box 2-6
Box Selection 7-4
breaking execution of commands 3-13

C

cell 2-1
cell domain 2-1
Cell Frame 9-13
Cell Information 4-21
cell instance 2-1
Cell Instance Selection 7-7
Cell Management 16-7
Cell Name Collision 4-16
cell origin 2-3
centerline 2-4
check box 3-12
check in 16-8
check out 16-8
child cell 2-1
CIF 4-7, 11-41
CIF file 2-8
circle 2-6
Circular Ring 5-12
Clip Out 12-30
Clipboard 8-3
clipboard of Expert 8-3
close project 4-11
Color Setup 3-14
column delta 2-6
column displacement 2-6
column index 2-6
Command bar 3-2
Command Line Input 17-120
compress 4-11
Configuration file 2-12
Coordinate Axes 12-5
Coordinate Origin 12-5
Coordinates 2-3
Copy 8-3
Copy Cells 4-15
Copy To Layer 7-6
Create cell 4-14
Create Cell in-Place 10-7
Creating Layout Objects 5-1
Cursor Crosshairs 12-5
Cursor Movement 3-10
Cursor Position 12-2
Curve Approximation 11-15
Custom Menu 3-14, 3-15, 11-46
customization 3-14
Cut 8-3
Cut by (Poly)Line 12-27

D

database unit 2-3
Deflection 11-15
Delete cells 4-17
Depth, of display of cell hierarchy 9-13
Derived Layer 11-25
Descend 4-13
Deselection 7-5
Design Bar 3-3
design plane 2-3, 3-2
Direct Pick 10-5
Direction Snap Modes 5-2
Discard Changes 4-21
Distance 12-3

Distance from Point to Line 12-5
Distance from Reference Point 12-6
donut 2-6, 5-12
drawing options 5-2
DRC 14-2
DRC Guard 14-5
DRC Script Execution Setup 11-50
DRC Script Running 14-2
Drop-down button 3-12
Drop-down list 3-12
Duplicate 8-5

E

ecf file 2-12
Edit box 3-12
Edit-in-Place 10-3
EIP 10-3
eld file 2-7
ellipse region 2-6
end style 2-4
Exclusive library 16-1
exclusive library 2-1
Expert Interface Scripts (XI-Shell) 17-1
explode 10-8
Exporting Cells 4-19
external technology file 4-6

F

filling 9-3
Filling Modes for Layers 9-5
Filling Modes for the Active Layer 9-5
Flat 9-11
flatten 10-9
Floorplan Mode 9-13

G

GDSII 4-7
GDSII file 2-8
GDSII I/O Setup 11-35
Geometric Transformations 8-5
Geometrical DRC System 14-2
Gravity 3-11
gravity mode 11-13
grid 2-7, 11-30
Guardian LPE Netlist Extractor 14-7
Guardian Netlist Comparator 14-9

H

Hierarchy Depth of Display 9-13
Hierarchy Operations 10-1
hole 5-3
Hole mode 5-4
hotkey 3-14, 11-44

I

Importing Cells 4-18
infix mode 11-12
instance (of a cell) 2-6
Instance /Array 5-6
instance position 2-1, 2-3
instancing 2-1
Integrated Layout Verification Tools 14-1
Introduction 1-1
Invert Selection 7-8

J

joint style 2-4

K

Keyboard Shortcut Definition File 2-12

L

label 2-6
layer 2-4
Layer Bar 3-6, 11-16
Layer Plan 11-17, 11-18
Layer Setup 11-15, 11-19
layout window 3-2
LayoutObject Type 17-106
Lazy 9-12
leaf cell 2-2
library 2-1
library file 2-1
list selection 7-5
loading project 4-5
logical library name 16-3

M

measurement 12-2
measurement unit 2-3, 2-7
Merge mode 5-4
merge mode 5-3
Metric Bar 3-4, 12-2
Minimal Grid 2-7
Minimal Grid (Grid Unit) 11-30
Mirror Operations 8-8
Modifying Layout Objects 6-1
Most Recent Used Project List 4-6
Move To Layer 7-7
Multiview Navigator 9-10

N

new cell 4-12
new project 4-3
Node Probing 14-10
Normal mode 5-3
Numeric Input Mode 5-3
numerical field 3-12

O

Offgrid 11-15
Open cell 4-13
Other Edit Submenu Operations 8-1
Over 5-5

P

Panning Commands 9-6
Parameterized Cell 2-3, 5-8
parent cell 2-1
Paste into Layer 8-4
P-cell 2-3, 5-8
Plot Preview 15-3
Plotter Setup 15-2
Plotting 15-1
Point Type 17-104
Position Cursor 8-4
Preselection Mode 7-3
primitive 2-4
project 2-1, 4-5
project file 2-1
Project/Cell Commands 4-1
PROPATTR 2-6
properties of objects 8-13
property map file 2-6
property name 2-6
property value 2-6

R

radiobutton 3-12
Read Only cell 4-21
Read Only mode of project loading 4-6
Rect Type 17-105
recursive instancing 2-2
Redo 8-9
Redraw 9-16
reference point 9-15
Reference Point, traveling 9-15
referencing 2-1
region 2-4
Region Modes 5-3
Rename cell 4-18
Restrictions for Layer Names 2-12, 11-21
rolled region 2-6, 5-12
rotate 8-7
rotate-90 8-7
row delta 2-6
row displacement 2-6
row index 2-6
ruler 12-4

S

Saving cells 4-21
screen layout 3-1
Search 8-15
search 8-15
SearchCriterion Type 17-107, 17-108
Security of Technology Settings 11-3
Selecting Layout Objects 7-1
selectoin modes 7-4
Set Origin 8-8
Setup 11-1
shape 2-4
Shared Access 4-6
Shared library 16-1
shared library 2-1
shortcut 3-14, 11-44
Snap Cursor to Grid 5-2
Snap Grid 2-7, 11-30
solid 9-3
stipple 9-3
Stretch 8-10
Stretching Modes 8-11
Supergrid 2-7, 11-31

T

tcn file 2-8
technology 11-3
technology file 2-11
Technology File from Dracula DRC Runset 11-2
technology file selection 4-4
text 2-6
To click 3-10
To double-click 3-10
To drag 3-10
To point 3-10
Tools 12-1
top cell 2-2
tree control 3-12

U

Undo 8-9
user defined property 2-6

User Interface 3-1

V

Version control 16-2
Viewing the Layout 9-1
Visual Grid 2-7, 11-31

W

width 2-4
wire 2-4
Wire Setup 11-24, 11-25
wireframe 9-3
work project 2-1

X

Xi Script Panel 17-120
Xi Script Submenu 12-6
XI-Command line 3-5
xi-script 17-1
xi-shell 17-1

Z

Zero-Width Wires 5-13
Zoom 9-8
  Zoom Out of Window 9-9
  Zoom Selected 9-8
  Zoom to Fit 9-8
  Zoom to Window 9-8